MATH EN INFO

TRAVAUX PRATIQUES N° 8

Plantons des choux

1 Objectif du TP

Maintenant que nous disposons d'une structure d'automate, nous allons (enfin) pouvoir apprendre à notre IA à réaliser des successions de tâches en respectant une certaine logique. Actuellement, notre IA n'a qu'un seul objectif, récolter du bois. C'est un peu limité. Nous allons lui apprendre à réaliser d'autres tâches et à décider par elle-même quand et comment réaliser ces nouvelles tâches.

2 Gestion de l'inventaire

Une des dernières données que notre IA ne prend pas encore en compte est son inventaire, or celui-ci va jouer un rôle évidemment important dans le mécanisme de prise de décision de notre IA. Nous allons donc commencer par le mettre en place. Il faut noter que l'inventaire d'Abigail ne contient pas des "Objets" au sens où nous les avons défini. En effet, dans notre modèle, les objets représentent des éléments présents sur la carte. Ce n'est pas le cas des objets de notre inventaire. Nous utiliserons donc un autre terme pour désigner les objets de l'inventaire, nous parlerons de ressources.

1. **A FAIRE : Dans le package "carte", créer un package "ressources" et à l'intérieur de ce nouveau package, créer une énumération "TypeRessource" contenant les éléments suivants : GOLD,BOIS,PARSNIPSEED, PARSNIPMATURE,CAULIFLOWERSEED,CAULIFLOWERMATURE, CHICKEN,EGG,MAYONNAISE ;**

Récolter un objet de la carte "produit" différentes ressources.

2. **A FAIRE : Ajouter à la classe "Objet" une méthode :**

   public abstract HashMap<TypeRessource,Integer> getLoot() ;

   et l'implémenter dans les classes filles. Attention pour les objets ne pouvant pas être récoltés (maison, escalier...) la méthode doit renvoyer une hashmap vide. Actuellement, seuls les arbres sont récoltables. Chaque arbre donne 12 unités de bois.

Il faut maintenant apprendre à notre IA à mémoriser les matériaux récoltés.

3. **A FAIRE : Ajouter au module mémoire, un attribut "inventaire" de type HashMap<TypeRessource,Integer> et l'initialiser à 0 pour chaque type de matériau dans le constructeur, sauf pour l'or que l'on initialisera à 500.**
4. **A FAIRE : Ajouter au module mémoire, une méthode**

   private void recolter(Objet objet)

   qui ajoute à l'inventaire les ressources obtenues en récoltant l'objet donné (attention à bien vérifier que l'objet est non nul).

5. **A FAIRE : Modifier la méthode "effectuerAction" du module mémoire pour gérer l'inventaire en cas d'action de type RECOLTE.**
6. **A FAIRE : Ajouter au module mémoire, une méthode**

   public int getQuantiteRessource(TypeRessource type)

   qui renvoie le nombre d'unités de ressources du type donné présentes dans l'inventaire.

3 Achetons des graines

La prochaine étape pour notre IA est de planter ses premières plantes (du panais). Cependant c'est un objectif complexe qui va nécessiter plusieurs étapes. En effet, Abigail doit vérifier que des graines de panais sont disponibles, aller les acheter, aller les planter, les arroser, les récolter, et les vendre... Heureusement, nous allons pouvoir profiter de la structure d'automate que nous avons mis en place à la séance précédente pour découper cette tâche compliquée en une succession de tâches simples. Nous allons donc grossir notre automate pour inclure ces nouvelles fonctionnalités.

7. **A FAIRE : En utilisant l'automate présenté à la fin du document, créer les classes des 7 nouveaux états de l'automate.**

3.1 Etape 1 : connaître les stocks du magasin

La première chose à faire pour Abigail est de demander l'état du magasin et de mémoriser cette information (qui change chaque jour). Commençons par apprendre à Abigail à mémoriser l'information.

8. **A FAIRE : Ajouter au module mémoire un attribut "stockMagasin" de type HashMap<TypeRessource,Integer> initialisé à null dans le constructeur.**
9. **A FAIRE : Ajouter et implémenter dans le module mémoire une méthode :**

   public void genererStockMagasin(int nbGrainePanais,int nbGraineChouFleur)

   qui génère le stock du magasin à l'aide des deux entiers donnés en paramètre.
10. **A FAIRE : Ajouter et implémenter dans le module mémoire les méthodes :**

    public boolean hasStockMagasin()

    public int getStockMagasin(TypeRessource type)

11. **A FAIRE : Modifier la méthode "effectuerAction" pour que dans le cas d'une action statique, "stockMagasin" soit mis à null.**

Apprenons maintenant à notre IA à demander l'information et à réagir à la réponse du serveur.

12. **A FAIRE : Ajouter "MAGASIN" à l'énumération TypeDemande.**
13. **A FAIRE : Modifier la méthode "creerDemande" de la fabriqueAction pour gérer cette nouvelle demande.**

14. **A FAIRE : Dans le module de réaction, créer une méthode :**

    private void reactionMagasin(String messageRecu)

    Implémenter cette méthode en utilisant la méthode .split("\\|") de la chaine de caractère "messageRecu" pour récupérer le nombre de graines de panais et de choux-fleurs disponibles et appeler la méthode "genererStockMagasin" du module mémoire.

15. **A FAIRE : Modifier la méthode "reagirAuMessageRecu" du module de réaction pour réagir à la nouvelle demande.**

Il ne nous reste plus qu'à faire que l'automate demande le stock du magasin s'il ne le possède pas.

16. **A FAIRE : Implémenter les deux méthodes de la classe "EtatDemandeMagasin" de telle sorte que :**
    - **action renvoie une nouvelle demande MAGASIN.**
    - **transition renvoie un nouvel EtatCheckAction.**
17. **A FAIRE : Implémenter les deux méthodes de la classe "EtatAcheter" de telle sorte que :**
    - **action renvoie null.**
    - **transition renvoie un nouvel EtatDemandeMagasin si le module mémoire ne connaît pas le stock du magasin et un nouvel EtatAllerDormir sinon.**
18. **A FAIRE : Modifier la méthode transition de "EtatCheckAction" pour qu'elle renvoie un nouvel EtatAcheter (à la place d'un nouvel EtatAllerVersArbre) dans le cas où la liste d'action est vide.**
19. **A FAIRE : Tester l'IA. Utiliser le mode débug pour vérifier que l'IA interprète bien la réponse du serveur.**

3.2 Etape 2 - Acheter des graines

Maintenant qu'Abigail connaît le stock du magasin, il lui faut aller acheter des graines. Pour cela nous allons commencer par localiser le magasin. Les deux cases de magasin pouvant être occupées par un arbre, on ne peut pas gérer le magasin comme on a géré les escaliers, nous allons donc procéder différemment.

20. **A FAIRE : Ajouter un attribut coordonneesMagasin de type "ArrayList<Coordonnee>" à la classe "Carte" ainsi que son assesseur.**
21. **A FAIRE : Modifier le constructeur de la classe carte pour qu'il initialise cette liste et y place les coordonnées des deux cases de magasin (voir le manuel du jeu).**

Il nous faut maintenant apprendre à notre IA à réaliser un achat.

22. **A FAIRE : Ajouter ACHAT à l'énumération TypeAction.**
23. **A FAIRE : Créer une nouvelle classe ActionAcheter héritant de la classe Action. Cette nouvelle classe disposera d'un attribut typeRessource de type TypeRessource que son constructeur initialisera à une valeur donnée en paramètre. (On achètera 1 graine).**
24. **A FAIRE : Ajouter et implémenter une méthode**

    public static Action creerActionAcheter(TypeRessource typeRessource)

    à la fabrique des actions.

Nous pouvons maintenant modifier l'automate pour apprendre à Abigail à aller acheter des graines.

25. **Ajouter un attribut booléen "vaAcheter" à la classe EtatAcheter et l'initialiser à faux.**
26. **Modifier la méthode "action" de EtatAcheter pour qu'elle suive le pseudo-code suivant :**

```
Si le module mémoire connaît le stock du magasin
        Si le joueur dispose d au moins 20 or et si le magasin possède
            au moins 1 unité de graine de panais.
                On crée un algorithme de Dijkstra et on lance les
                    calculs de distance depuis la position du joueur.
                On détermine laquelle des cases de magasin est la plus
                    proche.
                On passe vaAcheter à vrai.
                On se déplace jusqu à cette case.
                On ajoute une nouvelle action acheter du panais.
```

    On pourra s'inspirer de la méthode action de la classe EtatAllerVersArbre.
27. **A FAIRE : Modifier la méthode "transition" de cet état pour que dans le cas où les stocks du magasin sont connus, si vaAcheter est à vrai elle renvoie un nouvel EtatCheckAction et si vaAcheter est à faux elle renvoie un nouvel EtatAllerDormir.**
28. **A FAIRE : Tester votre IA.**

Vous pouvez constater que l'IA se rend bien au magasin et commence à acheter des graines... mais elle ne s'arrête pas de le faire et le serveur finit par renvoyer une erreur. Cela vient du fait que le module mémoire ne met pas à jour l'inventaire du joueur quand celui-ci achète des graines.

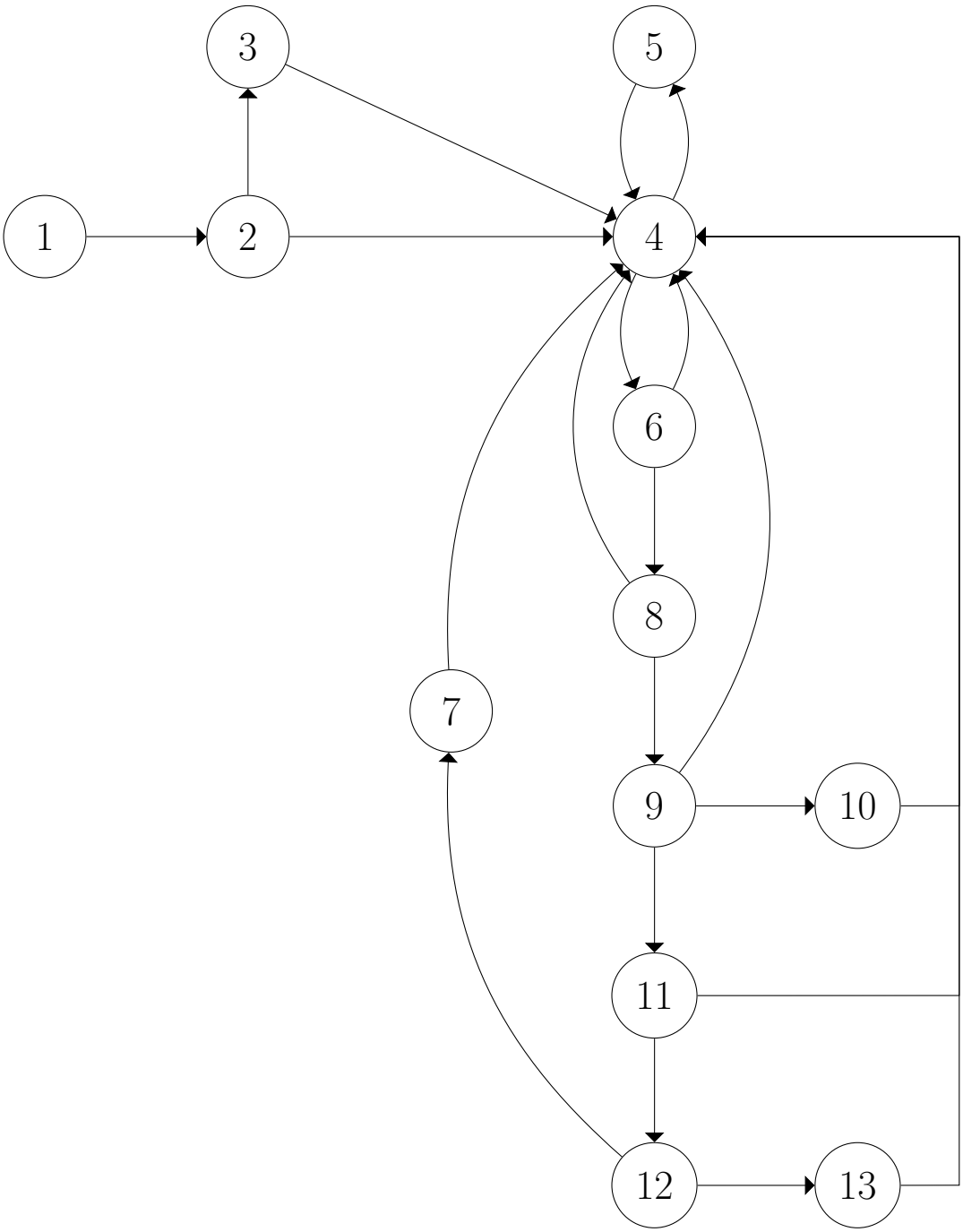
29. **A FAIRE : Modifier la méthode "effectuerAction" du module mémoire pour que lors d'une action ACHAT, elle retire 20 pièces d'or, ajoute 1 unité de graines de panais au joueur et retire 1 unité de graines de panais du magasin.**
30. **A FAIRE : Tester de nouveau l'IA.**

Afin de simplifier notre IA, nous avons implicitement limité notre IA à l'achat des graines de panais. Nous disposons de la plupart de ce qu'il faut pour permettre l'achat des graines de choux-fleurs (attention au coût des différents types de graines).

31. **BONUS : Ajouter une méthode abstraite "getTypeRessource" à la classe action pour que l'on puisse savoir quelle ressource est achetée.**
32. **BONUS : Modifier la méthode "effectuerAction" du module mémoire pour qu'elle gère les graines de choux-fleurs.**
33. **BONUS : Modifier la méthode "action" de EtatAcheter pour acheter des graines de choux-fleurs si on a déjà acheté toutes les graines de panais.**

Dans le prochain TP, nous implémenterons le reste de l'automate pour permettre à Abigail de (enfin) se lancer dans la production de masse de panais !

4 Description de l'automate



N°	Nom de la classe	Descriptif
1	EtatInitial	Point de départ de l'automate
2	EtatCheckCarte	Test si la carte existe
3	EtatDemanderCarte	Demande la carte au serveur
4	EtatCheckAction	Regarde si une action est déjà planifiée
5	EtatRealiserAction	Réalise l'action planifiée
6	EtatAllerRecolter	Détermine la plante la plus proche
7	EtatAllerDormir	Détermine comment rentrer pour dormir
8	EtatVendre	Va vendre ses légumes
9	EtatAcheter	Va acheter des graines
10	EtatDemandeMagasin	Demande le stock du magasin
11	EtatAllerPlanter	Va planter ses graines
12	EtatAllerArroser	Va arroser ses plantes
13	EtatAllerRemplir	Va remplir l'arrosoir