CHATEAUTOMATE

User documentation

Table of Contents

Introduction

Voici un guide utilisateur de notre jeu, Chateautomate. Celui-ci vous apprendra comment lancer le jeu, faire une (ou plusieurs) partie(s), ainsi que personnaliser le jeu.

Les automates

Le jeu repose sur une décision par automate. Vous devez donc les programmer avant de lancer le jeu via notre programme Ocaml dédié.

Action & Condition

L'autre aspect fondamental du jeu repose sur le fait qu'un automate effectue des actions en ayant vérifié des conditions. Cette mécanique forte simple engendre des restrictions et des choix stratégiques à effectuer sur vos automates, de manière à les rendre les plus performants possible !

Le jeu

Les automates codés précédemment permettent de contrôler le comportement d'une unité sur le terrain. De plus, cet automate se situe sur la carte de jeu sous forme de ville. Toute modification de cette ville engendrera une modification de l'automate, protégez-la à tout prix !

Comment jouer ?

Avant de lancer une partie, il vous faudra suivre une série d’étapes décrites ci-dessous :

Création d’un automate

La création d'un automate en OCaml visant à être utilisé par le programme principal se fait par le biais de l'assemblage de plusieurs plus petits automates, décrivant chacun une partie du comportement de l'automate final.

Afin de créer ces comportements, on pourra utiliser des fonctions décrivant des comportements prédéfinis (hostile, créateur,...), ou bien utiliser la fonction plus générale "transition" permettant de définir une transition grâce à ses composantes.

Ces comportements prennent tous au moins en paramètres un état courant et l'état suivant de l'automate, ce qui les rend compatibles avec la fonction appliquerSurListe qui permet d'appliquer une transition/un comportement sur une ou deux listes d'états, à la place d’un seul état courant/suivant.

Une fois la liste de ces comportements créées, il suffit d'utiliser la fonction creerAutomate sur cette dernière, qui renverra l'automate termine. Il suffit alors de rajouter dans la fonction principale un appel à ecrireXML, en donnant comme paramètres le nom du fichier XML voulu, ainsi que l'automate à traduire.

Des automates sont donnés en exemple dans createur.ml. Une fois que le fichier est complet, il suffit de compiler l’OCaml grâce à la commande *make*, puis de l’exécuter grâce à *./main*.

Liste de comportements prédéfinis et les paramètres nécessaires :

hostile *poids etat_courant etat_suivant* : attaque un ennemi adjacent quand il est dans *etat_courant*, puis passe dans *etat_suivant*

suiveur *poids etat_courant etat_suivant* : un automate qui suit un ordre qui lui a été donné si il est dans *etat_courant*, puis passe dans *etat_suivant*

recolteur *poids etat_courant etat_suivant* : un automate qui coupe un arbre adjacent quand il est dans *etat_courant*, puis passe dans *etat_suivant*

createur *poids type_unite cout etat_courant etat_suivant* : crée une unité *type_unite* s'il a au moins *cout* ressource

createurNbRatio *poids type_unite cout nb_min ratio_min etat_courant etat_suivant* : crée une unité *type_unite* s'il a assez de ressources, et qu'il y a au plus *nb_min* unités *type_unite*, ou qu'il y en a un ratio inferieur a *ratio_min*

createurZ *poids etat_courant etat_suivant* : comportement de zombie qui essaye de se dupliquer, puis passe dans *etat_suivant*

errant *poids etat_courant etat_suivant* : avance une fois au hasard quand dans *etat_courant*, puis passe dans *etat_suivant*

medecin *poids etat_courant etat_suivant* : soigne un allié adjacent quand il est dans *etat_courant* puis passe dans *etat_suivant*

constructeur *poids type_construction cout etat_courant etat_suivant* : construit un *type_construction* (« mur » ou « piege ») quand il a *cout* ressources

fonceur *poids etats_disponibles etat_courant etat_suivant* : choisi une direction au hasard, puis avance en la suivant jusqu'à rencontrer un obstacle. *etats_disponibles* doit être une liste de 4 états à laisser au comportement pour qu'il puisse être implémenté

chercheur *poids condition_sans_direction etats_disponibles etat_courant etat_suivant* : s'il trouve une case remplissant *condition_sans_direction*, il avance vers celle-ci, et attend une fois si jamais il perd de vue cette case. ***etats_disponibles*** a la même utilité que pour fonceur

Fonctions générales :

transition *poids condition action etat_courant etat_suivant* : crée un comportement à partir d'une seule transition en précisant chacune de ses composantes

appliquerSurListe *comportement etat_courant_liste etat_suivant_liste* : applique un comportement non plus sur un seul état courant et état suivant, mais sur une liste de ceux-ci. Pour une taille de *etat_courant_liste* de N et une taille de *etat_suivant_liste* de M, cela crée N*M *comportement*

creerAutomate *liste_comportement* : prend une liste d'automates *liste_comportement*, et crée un automate en regroupant les transitions de chaque élément

Création d’une classe

Pour crée une nouvelle classe, rendez-vous dans option, et créé votre classe personnalisée. Elle pourra être réutilisée dans la création d’unité.

Une classe possède différents attributs qui détermineront le coût de sa création dans le jeu. Voici une liste des caractéristiques détaillées :

- Deck Action : Ensemble d’action dont vous souhaitez doter votre classe.
- Deck Condition : Similaire au deck d’action, à la différence que celui-ci détermine quelle condition l’automate de la classe sera capable d’utiliser. Faite donc attention à bien les faire correspondre !
- Santé : Point de vie d’une unité.
- Attaque : Quantité de point de vie enlevé lors d’une attaque du personnage s’il possède l’action Attaqué.
- Puissance de soin : Point de vie rendu lors d’un soin du personnage (sur lui-même ou autrui) s’il possède l’action Soin.
- Armure : Réduction de dégâts appliqués lorsque le personnage subit une attaque.
- Bonus : Un attribut gratuit permettant de d’avantage spécialiser la classe.
    - Force : Davantage d’Attaque
    - Vie : Davantage de Point de Vie
    - Armure : Davantage d’Armure
- Traverse arbre : permet au personnage de traverser ou non les arbres sur la carte.

Création d’une unité

Une fois ces deux étapes franchies, vous pouvez créer l’unité de classe créé associée à l’automate que vous avez créé précédemment. Pour cela, cliquez sur Jouer.

Vous devrez donc sélectionner ces deux éléments dans les listes associées, et compléter la classe en lui donnant une apparence et des habits.

Lets play !

Deux modes de jeu s'ouvrent à vous à partir de là :

- Solo versus zombie : Pour ce mode, vous devez créer vos unités et les mettre en Joueur 1, puis créer une unité de classe Default et d'automate Zombie en Joueur 2. Ceci vous permettra de jouer contre une horde de zombies, qu'il vous faudra exterminer !
- Player versus Player : Dans ce jeu, il vous faudra trouver un partenaire qui remplira de la même manière que vous les emplacements d'unité du Joueur 2. Le but est d'être le dernier survivant ! Bonne chance !

Rajoutez des choses !

Le jeu comprend des fonctionnalités d'extension pour vous permettre de tester les choses par vous-même !

Les decks

Vous pouvez à loisir définir de nouveau deck d'action et de condition dans les dossiers adéquats. Vous les trouverez dans le dossier WorkShop à la racine du jeu. Pour ce faire, rajoutez simplement un fichier nommé comme vous souhaitez nommer votre deck, et inscrivez dedans les actions et conditions (selon le deck) que vous souhaitez avoir.

Action :

- Attaquer
- Attendre
- Avancer
- AvancerHasard
- AvancerJoueur(permet l'interaction clavier)
- Combattre (similaire à attaquer mais déclenche le combat en fenêtre latéral)
- ConstruireMur
- CouperBois
- Créer (permet de créer une unité d'un autre type)
- Dupliquer
- PoserPiege
- Raser
- Reparer
- Soigner

Condition :

- Ami
- ArbreProche
- CaseAmi
- Ennemi
- EnnemiProche
- Et
- Libre
- OrdreDonne (permet de savoir si un Joueur a donné un ordre au clavier)
- Ou
- RatioInf
- RessourcePossedees
- Type
- UnAmi
- UneCaseLibre
- UneCaseType
- UnEnnemi
- Vide

Ou et Et servent à permettre à un automate d'utiliser des conditions composées.

Cependant, attention ! Si vous choisissez trop de choses, vos automates seront très grands, et par conséquent la ville aussi !

Personnalisez vos classes

Les classes aussi peuvent être personnalisées en récupérant vos decks ajoutés ! Pour ce faire, au moment de créer une nouvelle classe, sélectionnez vos decks personnalisés pour les réutiliser.