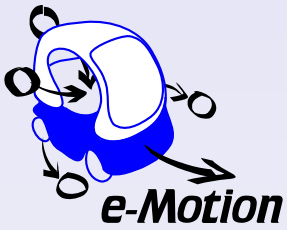
Programmation et apprentissage bayésien de comportements pour des personnages synthétiques

application aux personnages de jeux vidéos

Ronan Le Hy

sous la direction de Pierre Bessière

6 avril 2007



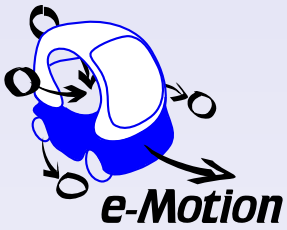
Unreal Tournament

[click me]

Fig.: Unreal Tournament : des personnages synthétiques et contrôlés par des humains s'affrontent dans une arène (image unrealtournament.com).

Enjeux

- pour le développeur : facilité, puissance
- pour le joueur : des comportements intéressants



Problèmes du point de vue du développeur

Programmer :

- facilité ?
- expressivité ?
- spécifier un comportement sans programmer ?
- besoins mémoire et processeur ?
- programmer/régler un comportement par l'exemple ? [Colin McRae Rallye 3]

Comprendre le programme :

- comprendre comment ajuster un comportement
- donner un sens aux réglages du comportement
- comprendre, modifier, maintenir un comportement complexe

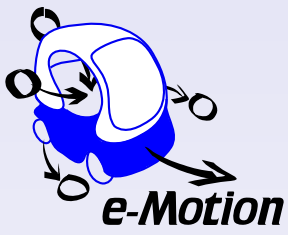
Problèmes du point de vue du joueur

Défauts :

- prédictible, répétitif ?
- rigide ?
- trop faible, trop fort ?

Défis :

- enseigner un comportement à un personnage (ami, ennemi, remplaçant) [Creatures], [Black and White], [Forza Motorsport]
- sauver, charger, échanger des comportements avec d'autre joueurs



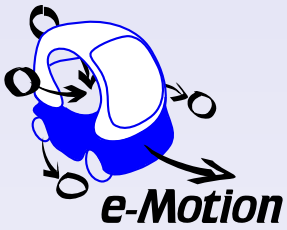
Questions essentielles

Paramètres :

- structure, localisation, identification
- intuition
- manipulation automatique et apprentissage

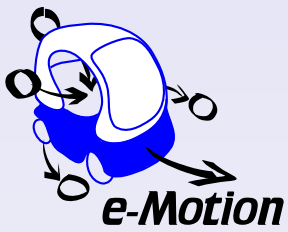
Ressources :

- mémoire
- processeur



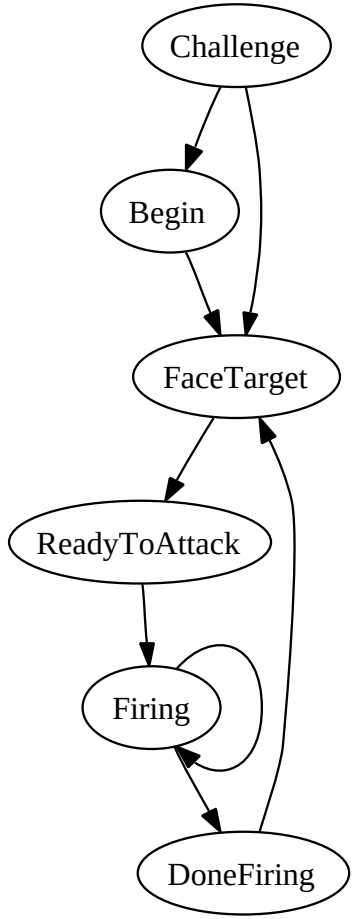
Plan

- Approche industrielle classique : Unreal Tournament
- Contributions
  - Construire une tâche : la fusion par cohérence améliorée
  - Séquencer des tâches : la programmation inverse
  - Apprendre un comportement
- Conclusion

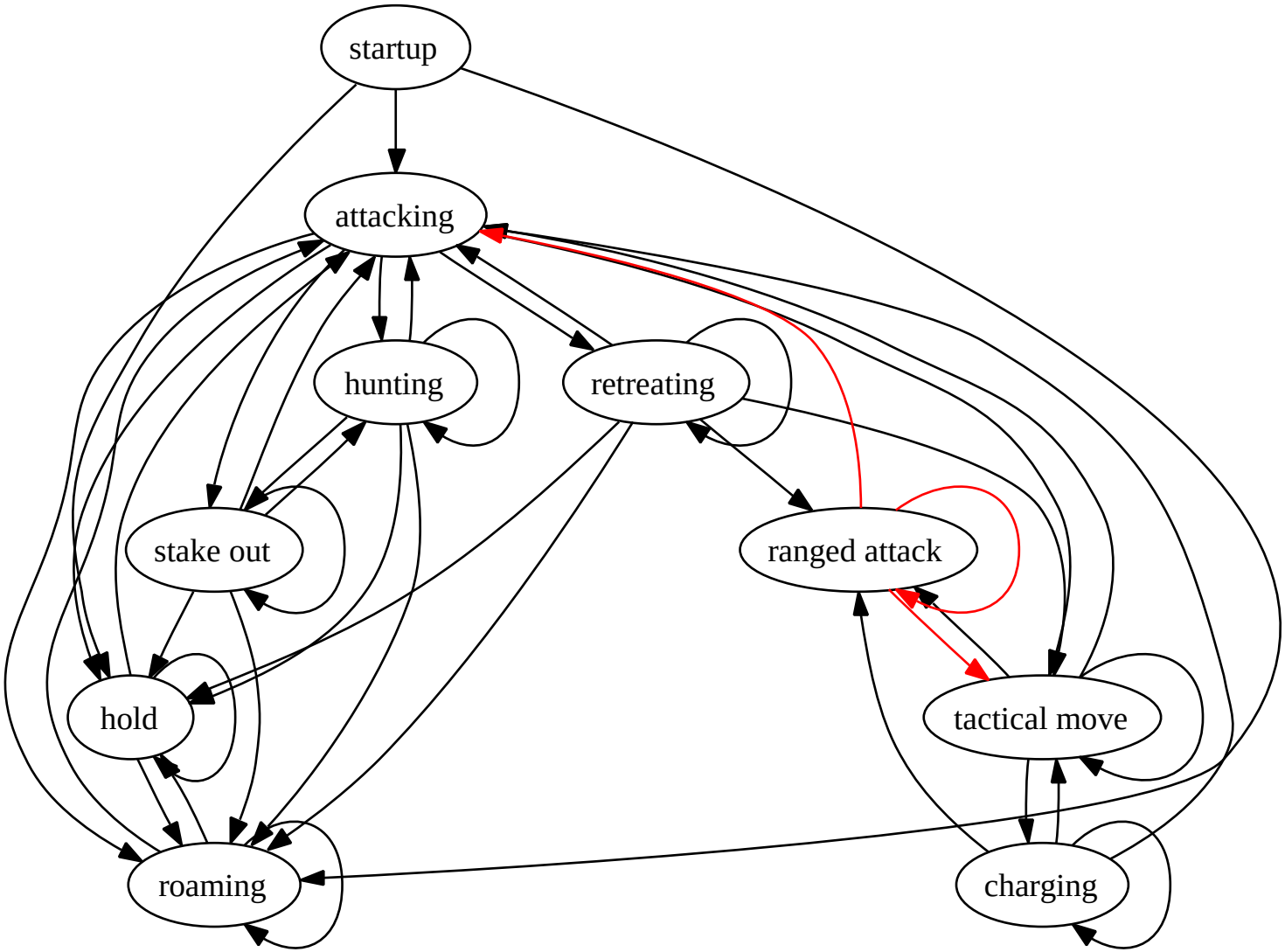


Méthode classique : une tâche, attaque à distance

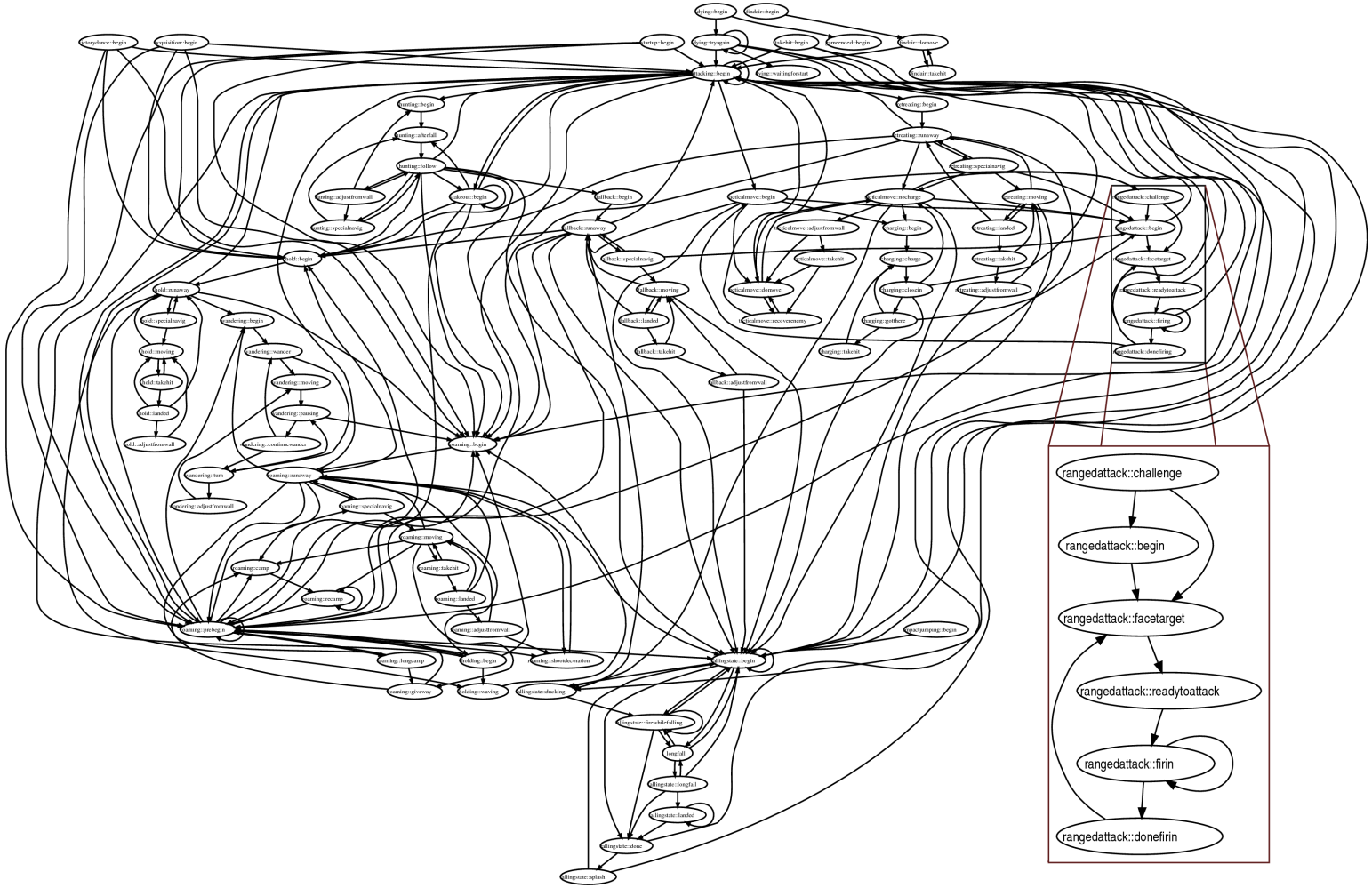
```
state RangedAttack {
// [...]
Challenge:
    Acceleration = vect(0,0,0); // stop
    DesiredRotation = Rotator(Enemy.Location - Location);
    PlayChallenge();
    FinishAnim();
    TweenToFighter(0.1);
    Goto('FaceTarget');
Begin:
    Acceleration = vect(0,0,0); // stop
    DesiredRotation = Rotator(Target.Location - Location);
    TweenToFighter(0.16 - 0.2 * Skill);
    // Goto('FaceTarget') implicite
FaceTarget:
    if ( NeedToTurn(Target.Location) )
    {
     PlayTurning();
     TurnToward(Target);
     TweenToFighter(0.1);
    }
    FinishAnim();
    // Goto('ReadyToAttack') implicite
ReadyToAttack:
    DesiredRotation = Rotator(Target.Location - Location);
    PlayRangedAttack();
    // Goto('Firing') implicite
Firing:
    TurnToward(Target);
    Goto('Firing');
DoneFiring:
    KeepAttacking();
    Goto('FaceTarget');
}
```



Méthode classique : une tâche de plus haut niveau, attaque

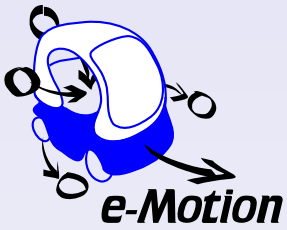


Méthode classique : comportement complet



Méthode classique : machines d'états finis, discussion

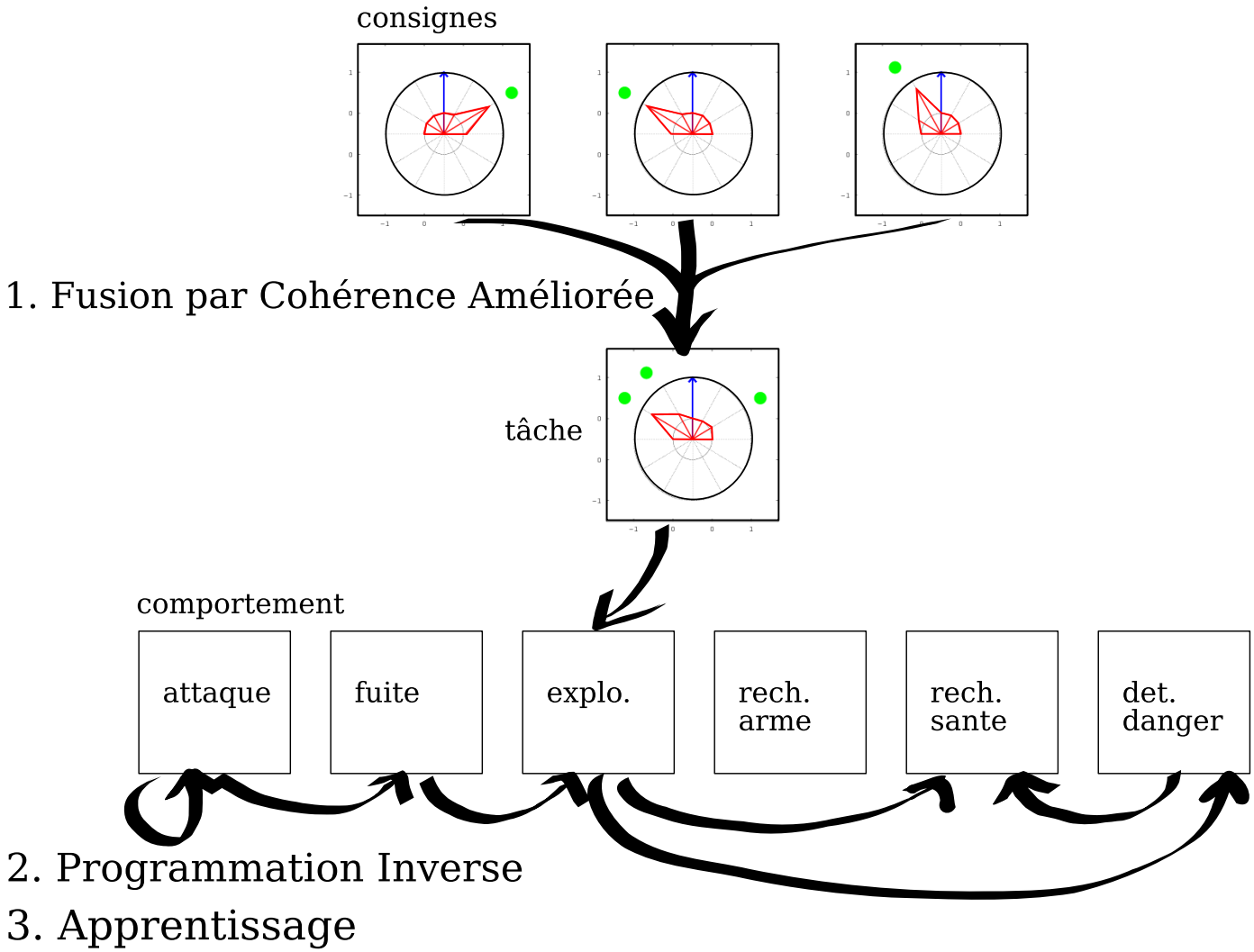
	paramètres				complexité		
	formalisme	structure	intuition	apprentissage	mémoire	processeur	non-spécialiste
UnrealScript	++	-	+	- -	++	++	- -



Autres approches pour la décision et la construction de comportements

- planification (foules : [Lamarche, Donikian 2004])
- machines d'états finis parallèles synchronisées [HPTS++, Lamarche]
- mise en compétition de tâches, arbitrage avec prise en compte de contraintes [INVIWO, Richard 2001]
- arbres de décision (apprentissage possible) [ID3, Quinlan 1975], [Hidden Markov decision trees, Jordan 1997]
- réseaux de neurones (apprentissage possible) [Colin McRae Rallye 3]

Contributions



Programmation bayésienne : filtre de Kalman

Programme

Description

Spécification

Variables pertinentes :

- S^0, ..., S^t : état de 0 à t
- O^0, ..., O^t : observation de 0 à t

Décomposition :

$$P(S^0\ O^0\ \dots\ S^t\ O^t) = P(S^0)\ P(O^0 \mid S^0)\ \Pi_{i=1}^{t} P(S^i \mid S^{i-1})\ P(O^i \mid S^i)$$

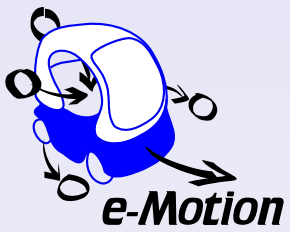
Formes paramétriques :

- $P(S^0) = G(S^0,\ \mu,\ \sigma)$
- $P(S^i \mid S^{i-1}) = G(S^i,\ A \times S^{i-1},\ Q)$
- $P(O^i \mid S^i) = G(O^i,\ H \times S^i,\ R)$

Identification :

Question(s) :

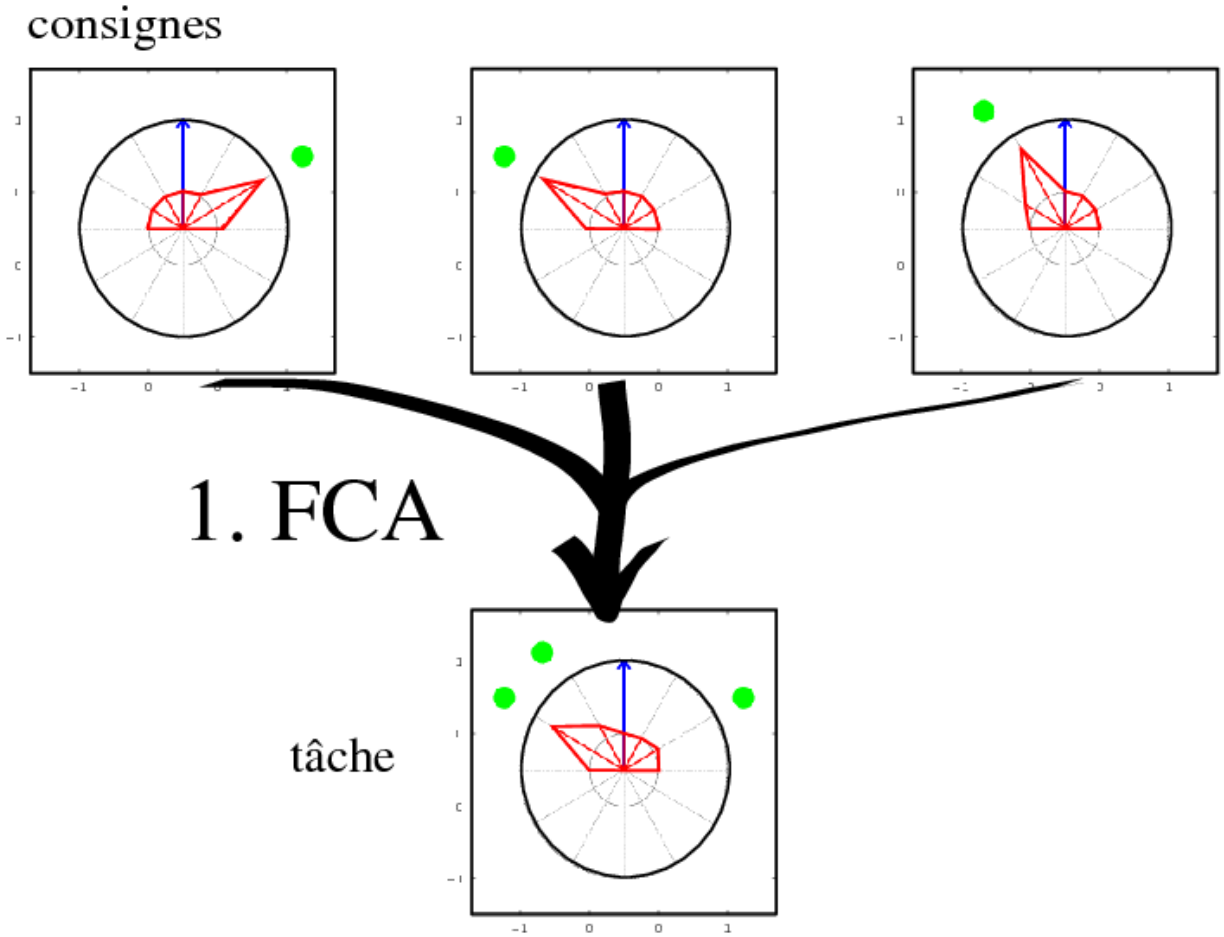
$$P(S^t \mid O^0 \dots O^t)$$



Programmation bayésienne

- formalisme générique de description de modèles probabilistes
- permet de séparer modélisation et inférence
- prend en compte l'incomplétude du modèle par la gestion de l'incertitude
- savoir faire important en programmation de systèmes autonomes ([Lebeltel 1999], [Diard 2003], [Garcia 2003], [Coué 2003], [Pradalier 2004], ...) et prise de décision ([Koike 2005], [Colas 2006])
- un problème transversal : combiner des programmes bayésiens

Fusion par cohérence améliorée : problématique [Pradalier Colas 2004]



Principe de fusion

Programme

Description

Spécification

Variables pertinentes :

- $\mathrm{V_{rot}}$: vitesse de rotation du personnage, dans $[\![-3; 3]\!]$
- $\mathrm{V^0_{rot}}$, $\mathrm{V^1_{rot}}$, ... $\mathrm{V^n_{rot}}$: consignes en vitesse de rotation
- $\mathrm{M^0}$, $\mathrm{M^1}$, ... $\mathrm{M^n}$: variables de cohérence pour chaque consigne, dans $0, 1$

Décomposition :

$$\mathrm{P(V_{rot}\ V^0_{rot}\ M^0\ V^1_{rot}\ M^1\ \dots\ V^n_{rot}\ M^n)} = \mathrm{P(V_{rot})} \prod_{i=0}^{n} \mathrm{P(V^i_{rot})\, P(M^i \mid V_{rot}\ V^i_{rot})}$$

Formes paramétriques :

- $\mathrm{P(V_{rot})}$: uniforme
- $\mathrm{P(V^i_{rot})}$: consigne issue d'un autre modèle
- $\mathrm{P([M^i = 1] \mid V_{rot}\ V^i_{rot}) = 1}$ si $\mathrm{v_{rot} = v^i_{rot}}$, 0 sinon

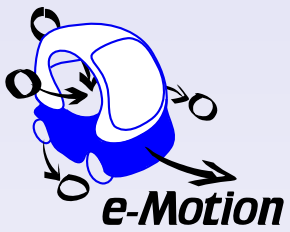
Identification :

Les consignes $\mathrm{P(V^i_{rot})}$ sont données dynamiquement par des sous-modèles, c'est-à-dire générée indépendamment au moment de chaque décision.

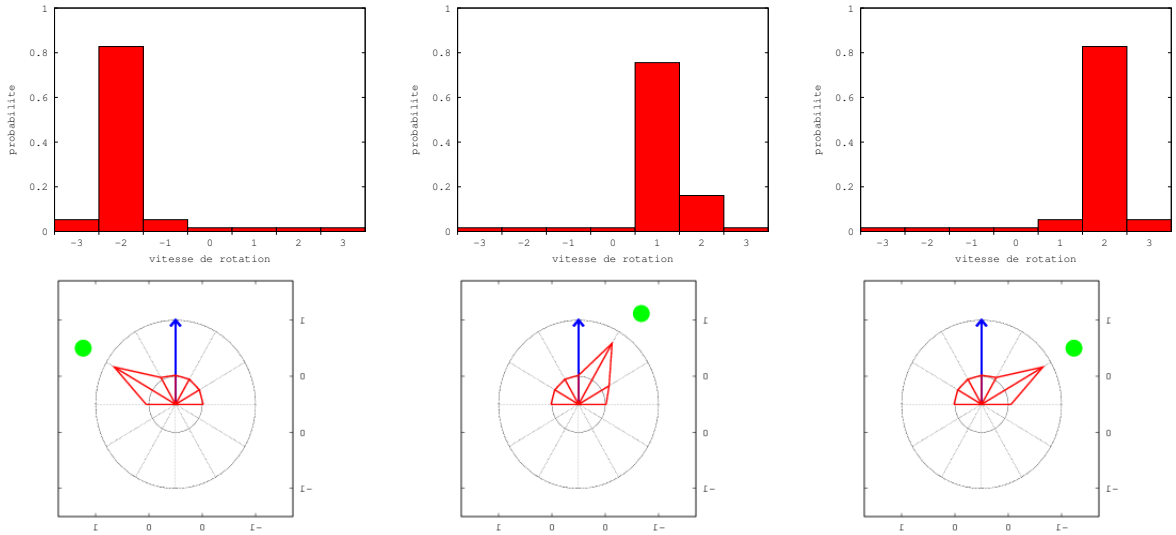
Question(s) :

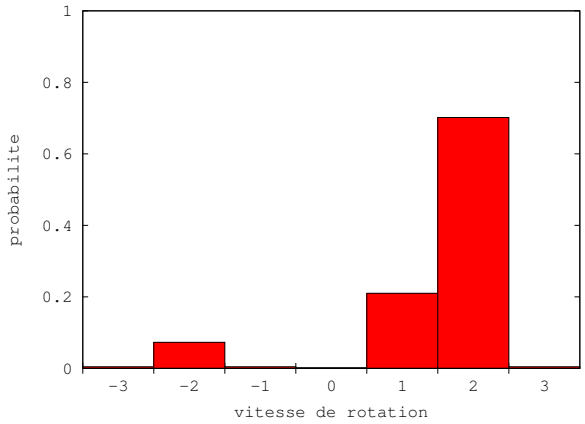
Question pour l'exécution du comportement :

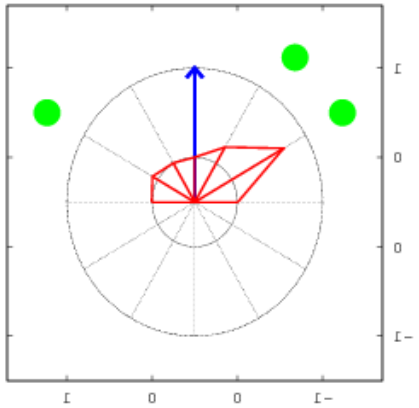
$$\mathrm{P(V_{rot} \mid M^0 \dots M^n)}$$



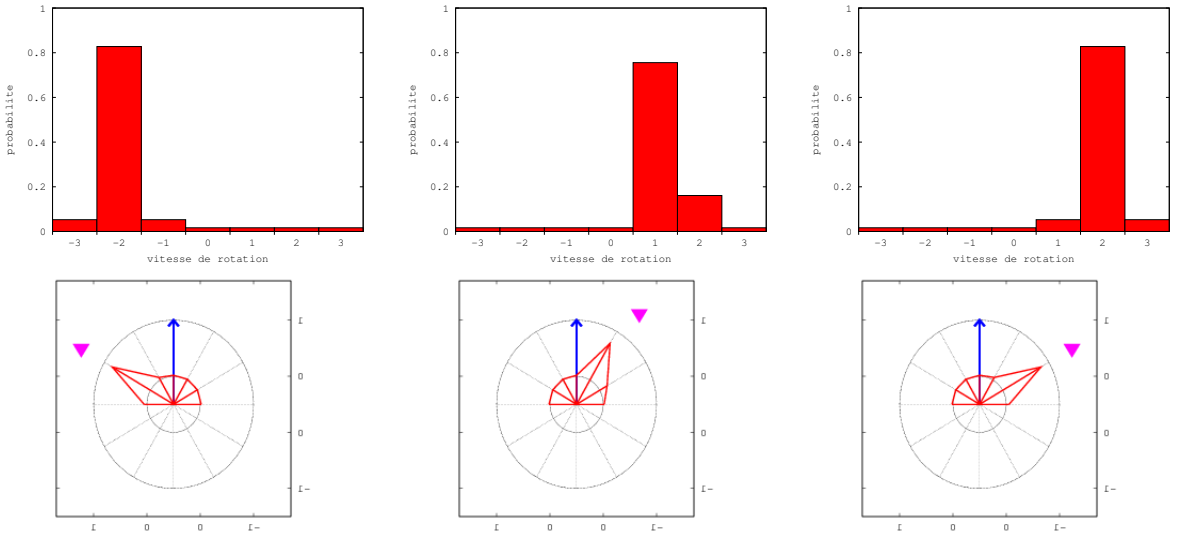
Fusion de consignes prescriptives : tâche d'exploration

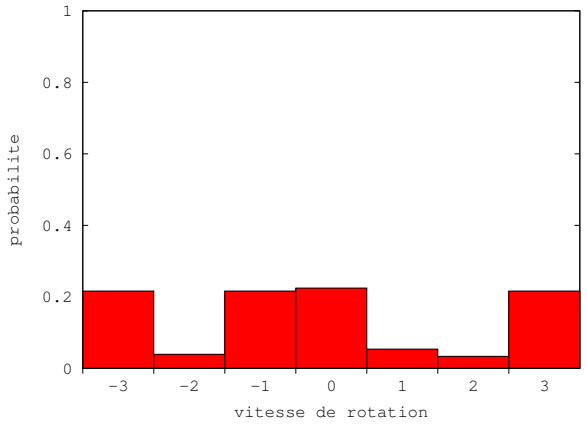


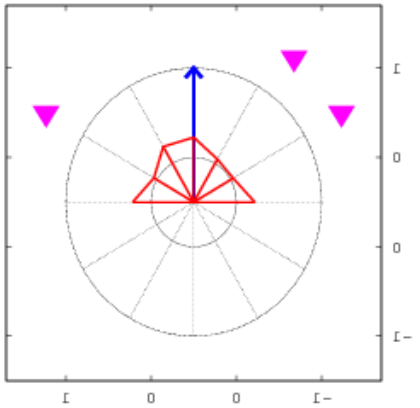




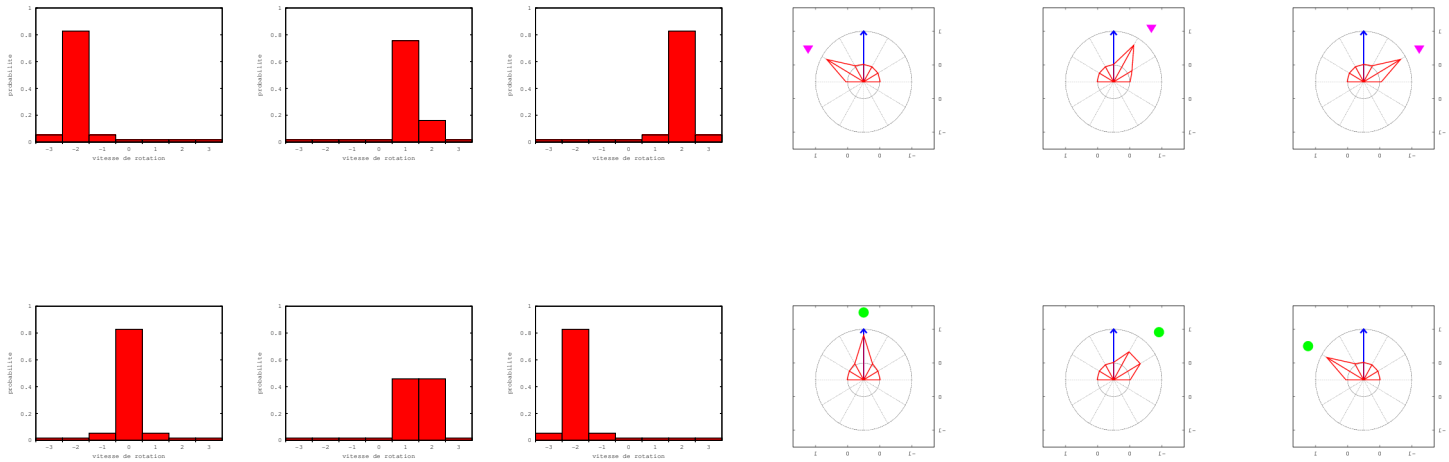
Fusion de consignes proscriptives : tâche de fuite

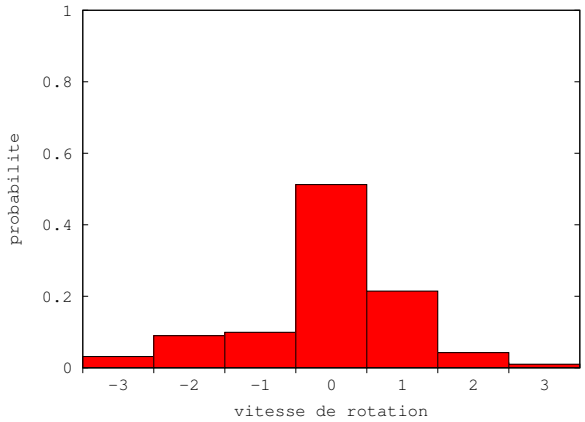


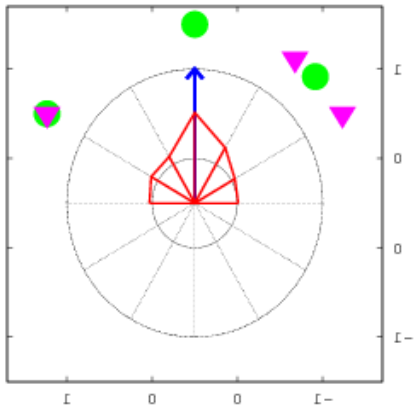




Fusion de consignes mélangées : tâche de fuite améliorée

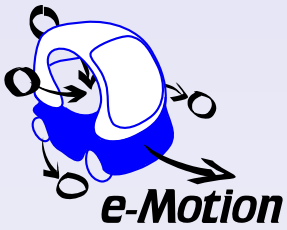




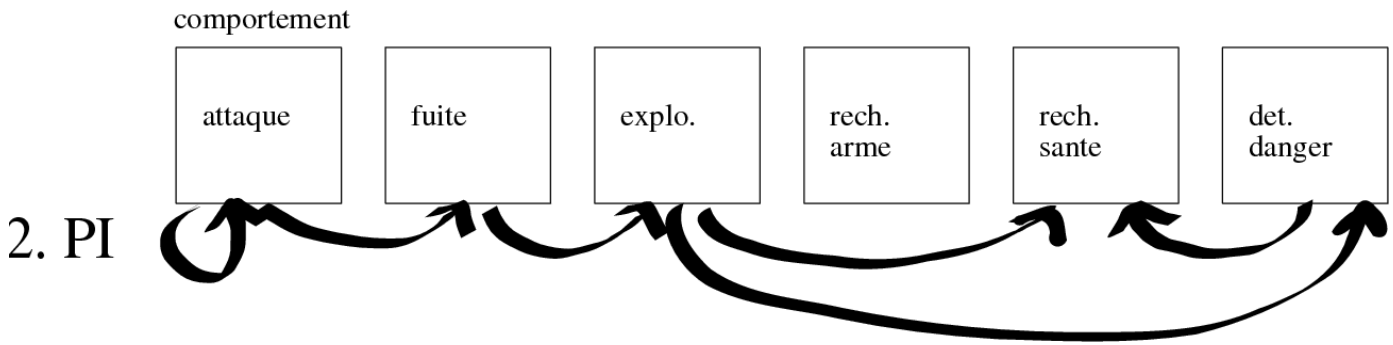


Fusion par cohérence améliorée : discussion

	paramètres				complexité		
	formalisme	structure	intuition	apprentissage	mémoire	processeur	non-spécialiste
UnrealScript	++	-	+	- -	++	++	- -
FCA	+	++	++	à venir	+	+	++



Programmation inverse : problématique



Programmation inverse : programme de séquencement

Programme

Description

Spécification

Variables pertinentes :

- tâches T^{t-1} et T^t, dans {Attaque, RArme, RSante, Explo, Fuite, DDanger}
- capteurs (tous à l'instant t) : {Sante, Arme, ArmeE, Bruit, NbE, ArmeProche, SanteProche}

Décomposition :

$$P(T^{t-1}\ T^t\ \text{Santé Arme} \ldots) = P(T^{t-1})\ P(T^t \mid T^{t-1})\ P(\text{Sante} \mid T^t)\ P(\text{Arme} \mid T^t)\ P(\text{ArmeE} \mid T^t)\ P(\text{Bruit} \mid T^t)\ P(\text{NbE} \mid T^t)\ P(\text{ArmeProche} \mid T^t)\ P(\text{SanteProche} \mid T^t)$$

Formes paramétriques :

- $P(T^{t-1})$: uniforme ;
- $P(T^t \mid T^{t-1})$: table de probabilités ;
- pour chaque Capteur, $P(\text{Capteur} \mid T^t)$: table de probabilités.

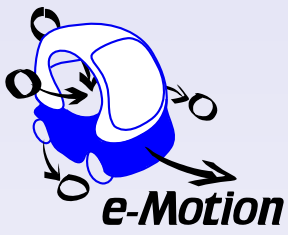
Identification :

$P(T^t)$ et $P(\text{Capteur} \mid T^t)$: tables de probabilités dont les paramètres peuvent être ajustés à la main, ou à l'aide de données expérimentales.

Question(s) :

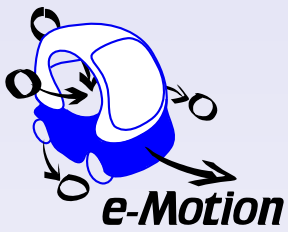
Question pour l'exécution du séquencement :

$$P(T^t \mid T^{t-1}\ \text{Santé Arme ArmeE Bruit NbE ArmeProche SantéProche})$$



Programmation inverse : caractéristiques

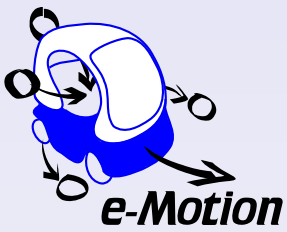
- hypothèse : capteurs indépendants entre eux sachant la tâche
- effets :
  - perte d'expressivité, mais
  - possibilité d'énumérer tous les cas possibles de manière compacte
  - inférence rapide
- programmation **inverse** :
  - `si` Arme `==Forte et` Sante `==Faible alors Goto('RechercheSanté')`
  - vs décrire $\mathrm{P}(\mathrm{Arme} \mid \mathrm{T}^{\mathrm{t}})$, $\mathrm{P}(\mathrm{Sante} \mid \mathrm{T}^{\mathrm{t}})$



La table de transition $P(T^t \mid T^{t-1})$

	T^{t-1}					
	Attaque	RArme	RSante	Explo	Fuite	DDanger
Attaque	0.95	0.01	0.01	0.01	0.01	0.01
RArme	0.01	0.95	0.01	0.01	0.01	0.01
RSante	0.01	0.01	0.95	0.01	0.01	0.01
Explo	0.01	0.01	0.01	0.95	0.01	0.01
Fuite	0.01	0.01	0.01	0.01	0.95	0.01
DDanger	0.01	0.01	0.01	0.01	0.01	0.95

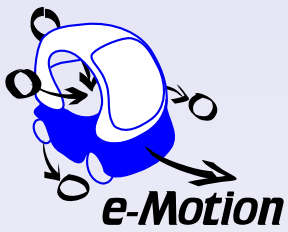
Tab.: $P(T^t|T^{t-1})$



La table $P(\mathrm{NbE} \mid \mathrm{T^t})$

	$\mathrm{T^t}$					
	Attaque	RArme	RSante	Explo	Fuite	DDanger
Aucun	0	0.89	0.33	0.9989	0	0.9989
Un	0.9	0.1	0.33	0.001	0.1	0.001
DeuxOuPlus	0.1	0.01	0.33	0.0001	0.9	0.0001

Tab.: $P(\mathrm{NbE}|\mathrm{T^t})$

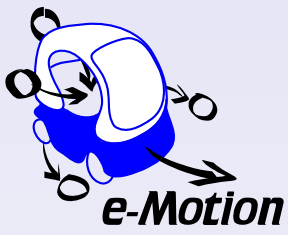


La table $P(\mathrm{NbE} \mid \mathrm{T}^t)$: versions prudente et agressive

	T^t					
prudent	Attaque	RArme	RSante	Explo	Fuite	DDanger
Aucun	0	0.89	0.33	0.9989	0	0.9989
Un	0.9	0.1	0.33	0.001	0.1	0.001
DeuxOuPlus	0.1	0.01	0.33	0.0001	0.9	0.0001

	T^t					
agressif	Attaque	RArme	RSante	Explo	Fuite	DDanger
Aucun	0	0.33	0.33	1.000	0.33	1.000
Un	0.2	0.33	0.33	10^{-9}	0.33	10^{-9}
DeuxOuPlus	0.8	0.33	0.33	10^{-10}	0.33	10^{-10}

Tab.: $P(\mathrm{NbE}|\mathrm{T}^t)$



Programmation inverse : résultats au combat

spécification manuelle, agressive	8.0
spécification manuelle, prudente	12.2
spécification manuelle, uniforme	43.2
personnage natif UT (niveau 3/8)	11.0

Tab.: Score entre 0 et 100, un score faible correspond à un personnage performant au combat.

Programmation inverse : discussion

	paramètres				complexité		
	formalisme	structure	intuition	apprentissage	mémoire	processeur	non-spécialiste
UnrealScript	++	-	+	- -	++	++	- -
FCA	+	++	++	à venir	+	+	++
PI	+	++	++	à venir	+	+	+

Apprentissage : problématique

Apprentissage par démonstration :

- le joueur joue
- les paramètres sont ajustés d'après les observations de son jeu

2 classes de problèmes :

- sans valeurs manquantes : comptage (tables de fréquences, Laplace), statistiques simples
- avec valeurs manquantes

Apprentissage avec contrôle par sélection de la tâche

Apprentissage avec sélection : résultats au combat

comportement	score
apprentissage par sélection de la tâche, agressive	45.7
spécification manuelle, agressive	8.0
spécification manuelle, prudente	12.2
spécification manuelle, uniforme	43.2
personnage natif UT (niveau 3/8)	11.0

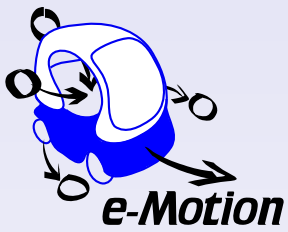
Tab.: Score entre 0 et 100, un score faible correspond à un personnage performant au combat.

Apprentissage avec heuristiques de reconnaissance de tâche

Apprentissage avec heuristiques : résultats au combat

comportement	score
apprentissage par reconnaissance avec heuristiques, agressive	4.4
apprentissage par reconnaissance avec heuristiques, prudente	13.9
apprentissage par sélection de la tâche, agressive	45.7
spécification manuelle, agressive	8.0
spécification manuelle, prudente	12.2
spécification manuelle, uniforme	43.2
personnage natif UT (niveau 3/8)	11.0

Tab.: Score entre 0 et 100, un score faible correspond à un personnage performant au combat.

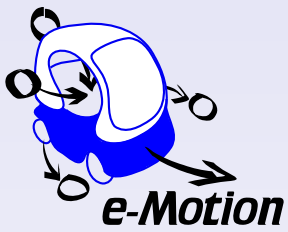


Algorithme de Baum-Welch [Baum 1972] [Rabiner 1989]

Algorithme [Expectation-Maximization] : alterner entre

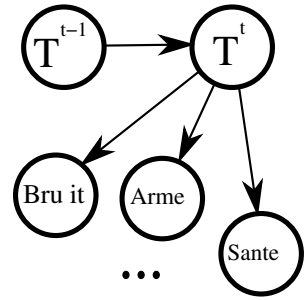
- (E) distribution sur les tâches sachant les observations et les paramètres
- (M) paramètres sachant les observations et la distribution sur les tâches

Initialement : les paramètres (tables) sont fixés arbitrairement.
Baum-Welch : la structure de HMM permet une implantation rapide des deux phases.
Baum-Welch incrémental : E, M à chaque observation.



Baum-Welch sur le modèle de séquencement

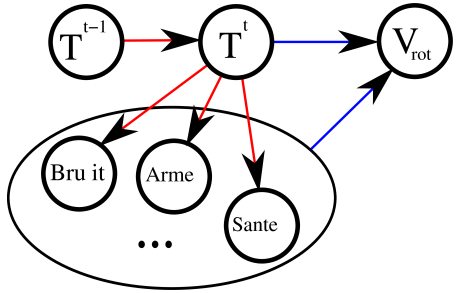
$$P(T^{t-1}\ T^{t}\ \text{Bruit}\ \text{Arme}\ \text{Sante}\ \ldots) =$$
$$P(T^{t-1})$$
$$P(T^{t} \mid T^{t-1})$$
$$P(\text{Bruit} \mid T^{t})\ P(\text{Arme} \mid T^{t})\ P(\text{Sante} \mid T^{t})\ \ldots$$



$$(\text{E}) :\ P(T^{t-1}\ T^{t} \mid \text{Bruit}\ \text{Arme}\ \text{Sante}\ \ldots\ \theta^{n-1}) = \frac{1}{Z} \times$$
$$P(T^{t-1}\ \theta^{n-1})$$
$$P(T^{t} \mid T^{t-1}\ \theta^{n-1})$$
$$P(\text{Bruit} \mid T^{t}\ \theta^{n-1})\ P(\text{Arme} \mid T^{t}\ \theta^{n-1})\ P(\text{Sante} \mid T^{t}\ \theta^{n-1})\ \ldots$$

Baum-Welch sur le modèle complet tâches-séquencement

$$
\begin{aligned}
&\mathrm{P}(\mathrm{T}^{t-1}\ \mathrm{T}^{t}\ \mathrm{Bruit}\ \mathrm{Arme}\ \mathrm{Sante}\ \ldots\ \mathrm{V_{rot}}) = \\
&\quad \mathrm{P}(\mathrm{T}^{t-1}) \\
&\quad \mathrm{P}(\mathrm{T}^{t} \mid \mathrm{T}^{t-1}) \\
&\quad \mathrm{P}(\mathrm{Bruit} \mid \mathrm{T}^{t})\, \mathrm{P}(\mathrm{Arme} \mid \mathrm{T}^{t})\, \mathrm{P}(\mathrm{Sante} \mid \mathrm{T}^{t}) \ldots \\
&\quad \mathrm{P}(\mathrm{V_{rot}} \mid \mathrm{T}^{t}\ \mathrm{Bruit}\ \mathrm{Arme}\ \mathrm{Sante}\ \ldots)
\end{aligned}
$$



$$
\begin{aligned}
&(\mathrm{E}) :\ \mathrm{P}(\mathrm{T}^{t-1}\ \mathrm{T}^{t} \mid \mathrm{Bruit}\ \mathrm{Arme}\ \mathrm{Sante}\ \ldots\ \mathrm{V_{rot}}\ \theta^{n-1}) = \frac{1}{\mathrm{Z}} \times \\
&\quad \mathrm{P}(\mathrm{T}^{t-1}\ \theta^{n-1}) \\
&\quad \mathrm{P}(\mathrm{T}^{t} \mid \mathrm{T}^{t-1}\ \theta^{n-1}) \\
&\quad \mathrm{P}(\mathrm{Bruit} \mid \mathrm{T}^{t}\ \theta^{n-1})\, \mathrm{P}(\mathrm{Arme} \mid \mathrm{T}^{t}\ \theta^{n-1})\, \mathrm{P}(\mathrm{Sante} \mid \mathrm{T}^{t}\ \theta^{n-1}) \ldots \\
&\quad \mathrm{P}(\mathrm{V_{rot}} \mid \mathrm{T}^{t}\ \mathrm{Bruit}\ \mathrm{Arme}\ \mathrm{Sante}\ \ldots)
\end{aligned}
$$

Apprentissage bayésien : résultats au combat

[click me]

comportement	score
apprentissage bayésien, démonstration agressive	8.5
apprentissage par reconnaissance avec heuristiques, agressive	4.4
apprentissage par reconnaissance avec heuristiques, prudente	13.9
apprentissage par sélection de la tâche, agressive	45.7
spécification manuelle, agressive	8.0
spécification manuelle, prudente	12.2
spécification manuelle, uniforme	43.2
personnage natif UT (niveau 3/8)	11.0

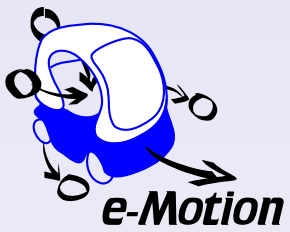
Tab.: Score entre 0 et 100, un score faible correspond à un personnage performant au combat.

Apprentissage bayésien : tables apprises

	T^t					
prudent	Attaque	RArme	RSante	Explo	Fuite	DDanger
Aucun	0	0.89	0.33	0.9989	0	0.9989
Un	0.9	0.1	0.33	0.001	0.1	0.001
DeuxOuPlus	0.1	0.01	0.33	0.0001	0.9	0.0001

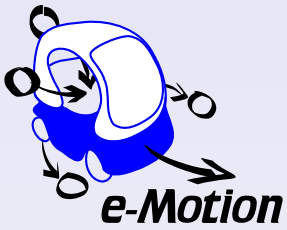
NbE \ T^t	Attaque	RArme	RSante	Explo	Fuite	DDanger
Aucun	0.053	0.97	0.011	0.7	0.0013	0.1
Un	0.54	0.018	0.51	0.14	0.52	0.62
DeuxOuPlus	0.41	0.0117	0.478	0.159	0.48	0.277

Tab.: $P(\mathrm{NbE}|\mathrm{T}^t)$: versions manuelle et apprise



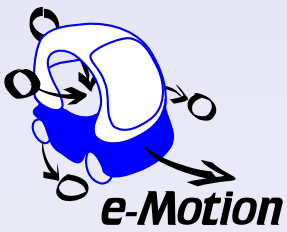
Apprentissage de comportements : discussion

	paramètres				complexité		
	formalisme	structure	intuition	apprentissage	mémoire	processeur	non-spécialiste
UnrealScript	++	-	+	- -	++	++	- -
FCA	+	++	++	++	+	+	++
PI	+	++	++	++	+	+	+



Contributions

- fusion par cohérence améliorée
- programmation inverse
- modèles de comportements appris par démonstration



Réponse aux problèmes du développeur

Programmer :

- facilité ?
- ✓expressivité ?
- ✓spécifier un comportement sans programmer ?
- ✓besoins mémoire et processeur ?
- ✓programmer/régler un comportement par l'exemple ?

Comprendre le programme :

- ✓comprendre comment ajuster un comportement
- ✓donner un sens aux réglages du comportement
- comprendre, modifier, maintenir un comportement complexe

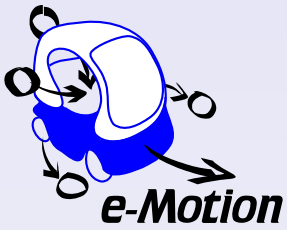
Réponse aux problèmes du joueur

Défaults :

- prédictible, répétitif ?
- rigide ?
- trop faible, trop fort ?

Défis :

- ✓enseigner un comportement à un personnage (ami, ennemi, remplaçant)
- ✓sauver, charger, échanger des comportements avec d'autre joueurs



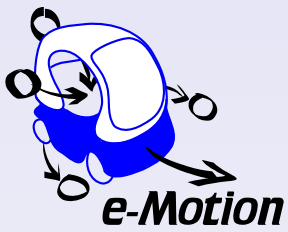
Perspectives

Outils de construction en programmation bayésienne :

- apprentissage de séquencements hiérarchisés
- enrichissement de l'état : tâche + ?
- prendre plus d'un pas de temps en arrière pour l'état
- fusion par cohérence améliorée pour le séquencement

Passage à l'échelle :

- application à un monde virtuel plus riche
- implémentation industrielle



Merci

Fig.: (image Francis Colas)

Algorithme de Baum-Welch : réestimation incrémentale des paramètres

notations : paramètres θ, observations O de 0 à n

estimation de la probabilité d'être dans la tâche i au vu de $\mathrm{O}^0 \ldots \mathrm{O}^\mathrm{n}$:

$$\widetilde{\mathrm{P}}^\mathrm{n}([\mathrm{T} = \mathrm{i}]) = \frac{\sum_{\mathrm{t}=0}^{\mathrm{n}} \mathrm{P}([\mathrm{T}^{\mathrm{t}-1} = \mathrm{i}] \mid \theta^\mathrm{t})}{\mathrm{n}+1}$$

estimation de θ^n sachant O^n et $\theta^{\mathrm{n}-1}$:

$$\mathrm{P}([\mathrm{T}^\mathrm{t} = \mathrm{j}] \mid [\mathrm{T}^{\mathrm{t}-1} = \mathrm{i}]\, \theta^\mathrm{n}) =$$
$$\frac{1}{\mathrm{Z}'} \times \left(\mathrm{P}([\mathrm{T}^\mathrm{t} = \mathrm{j}] \mid [\mathrm{T}^{\mathrm{t}-1} = \mathrm{i}]\, \theta^{\mathrm{n}-1}) + \frac{\mathrm{P}([\mathrm{T}^{\mathrm{n}-1} = \mathrm{i}]\, [\mathrm{T}^\mathrm{n} = \mathrm{j}] \mid \theta^{\mathrm{n}-1})}{\mathrm{n}\, \widetilde{\mathrm{P}}^{\mathrm{n}-1}([\mathrm{T} = \mathrm{i}])} \right)$$

$$\mathrm{P}([\mathrm{O}^\mathrm{t} = \mathrm{k}] \mid [\mathrm{T}^{\mathrm{t}-1} = \mathrm{j}]\, \theta^\mathrm{n}) =$$
$$\frac{1}{\mathrm{Z}''} \times \left(\mathrm{P}([\mathrm{O}^\mathrm{t} = \mathrm{k}] \mid [\mathrm{T}^{\mathrm{t}-1} = \mathrm{j}]\, \theta^{\mathrm{n}-1}) + \frac{\mathrm{P}([\mathrm{O}^\mathrm{n} = \mathrm{k}]\, [\mathrm{T}^\mathrm{n} = \mathrm{j}] \mid \theta^{\mathrm{n}-1})}{\mathrm{n}\, \widetilde{\mathrm{P}}^{\mathrm{n}-1}([\mathrm{T} = \mathrm{j}])} \right)$$