Planification dans les jeux-vidéos : À quoi servent les coûts des actions ? *

Guillaume Prévost[1,2] Stéphane Cardon[1] Tristan Cazenave[2] Éric Jacopin[1]

[1] CREC Saint-Cyr, Écoles de Coëtquidan, F-56381 GUER
[2] LAMSADE - Université Paris-Dauphine, F-75775 Paris Cedex 16

guillaume.prevost@st-cyr.terre-net.defense.gouv.fr

Résumé

Nous discutons de l'intérêt des coûts des actions dans le planificateur G.O.A.P. (Goal-Oriented Action Planning), qui est utilisé pour le contrôle en temps-réel des personnages non-joueurs dans certains jeux-vidéos. Nous avons instrumenté l'activité de planification du jeu vidéo F.E.A.R. afin d'étudier l'usage des coûts des actions par le planificateur. Nous montrons et nous expliquons pourquoi leur impact est faible en pratique.

Mots Clef

Planification d'actions, A*, Coût des actions, Jeux-vidéo

Abstract

This paper suggests a reflection on the costs of the actions used by the planner G.O.A.P. (Goal-Oriented Action Planning) which is used in some video games like F.E.A.R. to have real-time control of the non-playable character. We have analysed the planning activity of F.E.A.R. to study the utility of the costs in the planner. We show and explain why they have little impact in practice.

Keywords

Actions planning, A*, Actions cost, Video Game.

1 Introduction

La planification d'actions en temps-réel pour la conception de jeux vidéo a pour avantage de découpler les actions des différents états possibles d'un personnage non-joueur (PNJ). La conception des actions est ainsi facilitée car leur ajout ou leur retrait ne nécessitent plus de modifier l'intégralité du système. Le couplage se fera en jeu, par le planificateur, qui générera des plans en chainant les actions candidates. Le système G.O.A.P. (Goal-Oriented Action Planning), créé en 2003 [5] et mis en œuvre pour la première fois en 2005 dans le jeu vidéo F.E.A.R. par Jeff Orkin [6], est un algorithme de planification basé sur celui de `STRIPS` [2]. Il est utilisé dans de nombreux jeux et notamment dans "Shadow of Mordor" [4] ou encore "Rise of the Tomb Raider" [1]. Il fonctionne de la manière suivante : les buts d'un PNJ représentent des états souhaités du monde (états buts) ; les différents états du monde sont modélisés par des symboles ; chacun des PNJs possède des capteurs (vision, vie, sons...) et une mémoire afin de percevoir et de comprendre le milieu où ils se trouvent ; les actions possèdent un coût et sont équipées de *préconditions* et de *postconditions* (ou *effets*) qui ne sont, ni plus ni moins, que des symboles d'état du monde ; enfin, la génération de plans est faite grâce à l'algorithme de plus court chemin A* [3], qui est déjà mis en œuvre pour la recherche de trajectoire pour le déplacement des PNJs. Nous avons fait finir le jeu F.E.A.R. à trois personnes différentes et au moins deux fois. L'une des deux parties était configurée avec tous les coûts d'action mis à 1 tandis que la deuxième se déroulait avec les coûts d'origine. L'étude des données expérimentales récoltées durant ces sessions montre que l'ensemble des plans d'actions générés dans la configuration des coûts originaux (CO) sans l'être dans celui des coûts unitaires (CU) est faible à 5,14%. Ainsi, dans G.O.A.P., à quoi servent les coûts des actions ? Le premier argument avancé est qu'il permet de forcer un choix dans le planificateur. Entre autres, dans F.E.A.R., plus une action sera spécifique plus son coût sera faible afin de la favoriser. Par exemple, les actions *Attack* et *AttackFromCover* ont le même effet mais leur coût diffère avec une valeur de 6.0 pour la première et de 4.0 pour la deuxième. Il y a 18 actions différentes qui possèdent pour seul et unique effet celui d'*Attack* et d'*AttackFromCover* et leur coût varie de 1.0 à 7.0 (Certaines ont le même coût). C'est la catégorie la plus variée. L'autre raison pour l'existence des coûts est que cela permet la réutilisation de A* qui est mis en œuvre pour le déplacement des PNJs. Pour ce faire, l'algorithme de plus court chemin a besoin d'une fonction heuristique et d'une fonction de coûts pour sommer les deux résultats afin de choisir le meilleur candidat à la frontière du graphe de recherche. Soit "n" un état du monde intermédiaire entre l'état initial et l'état but, sachant que la recherche de plans commence depuis l'état but pour atteindre l'état initial, la fonction heuristique évalue le nombre de symbole d'état du monde à satisfaire entre "n" et l'état ini-

*Les travaux décrits sont financés par SAFRAN dans le cadre de la chaire « Soldat Augmenté ».

tial tandis que la fonction de coût est la somme du coût des actions nécessaires pour atteindre "n" depuis l'état but. Par exemple, imaginons un PNJ qui n'a pas son arme chargée et qui reçoit le but *KillEnemy*. Le symbole de l'état du monde *TargetIsDead* doit avoir la valeur *true*. La fonction coût de l'état but est nulle car aucune action n'est choisie pour l'instant tandis que sa fonction heuristique vaut 1.0 puisqu'un symbole est non-satisfait. Les attaques *Attack* et *AttackMelee* sont proposées au planificateur puisque leur effet satisfait le symbole d'état du monde. Les préconditions de *AttackMelee* sont toutes satisfaites alors qu'*Attack* a besoin du symbole *WeaponLoaded* mis à *true*. *AttackMelee* qui a un coût de 3.0 pointe sur un état du monde qui n'a pas de symbole insatisfait, il s'agit de l'état initial, donc la fonction heuristique vaut 0.0 et sa fonction d'évaluation vaut 3.0 (3.0 + 0.0). *Attack* qui a un coût de 6.0 pointe sur un état du monde qui a un symbole insatisfait, donc sa fonction heuristique vaut 1.0 et sa fonction d'évaluation vaut 7.0. Imaginons que cette dernière action soit choisie (Ce qui n'arrivera pas en pratique car 3.0 < 7.0), l'action *Reload* sera proposée, elle a un coût de 5.0 et pointe sur l'état initial. L'état pointé a donc une fonction heuristique qui vaut 0.0, une fonction de coût qui vaut 11.0 (6.0 + 5.0) et, finalement, une fonction d'évaluation qui vaut 11.0. La suite du papier est organisée de la façon suivante : un premier paragraphe présentant tous les résultats obtenus avec les figures fournies en annexe et un second paragraphe exposant les discussions et les conclusions.

2 Résultats

Sur les six sessions (une session représentant le jeu en entier) avec les deux configurations, 270.866 plans ont été générés : 139.939 plans pour les coûts originaux(CO) avec 19h59 de temps de jeu et 130.927 plans pour les coûts unitaires(CU) avec 18h18 de temps de jeu. Parmi ces plans, 83 sont différents dans la configuration CO et 68 dans la configuration CU. 65 sont en commun et ils représentent 94,86% des plans pour CO et 99,96% des plans pour CU. Les plans générés avec la configuration CU sont donc inclus dans ceux de la configuration CO. La diversité engendrée par les coûts originaux représente 5,14%. La Figure 2 montre que la probabilité d'apparition d'un plan, en fonction du nombre d'actions le composant, est la même, que les coûts soient originaux ou que les coûts soient unitaires. Les échecs représentent respectivement 74,05% et 76,88% des plans générés. Un échec correspond à l'incapacité du planificateur à retourner au moins un plan valide pour résoudre un but. C'est le cas lorsque qu'aucune action n'est candidate ou lorsqu'au moins une précondition de chaque action candidate ne peut être satisfaite. Ensuite, 103.626 plans sont réussis pour CO : 68,32% possèdent une seule action, 30,50% possèdent deux actions, 0,86% en possèdent trois et 0,32% en possèdent quatre. Pour CU, 130.927 plans sont réussis : 66,45% n'ont qu'une seule action, 32,49% en ont deux, 0,79% en ont trois et 0,27% en ont quatre. La majeure partie des plans n'ont donc qu'une ou deux actions puisqu'ils représentent 98,9% des plans générés réussis dans les deux configurations. La Figure 1, qui compare la probabilité d'apparition des 58 plans en commun entre les deux configurations, nous permet d'observer que l'influence des coûts des actions est majoritairement négligeable sur le comportement du planificateur de F.E.A.R.. Les plans d'actions ont une probabilité d'apparition équivalente pour chaque configuration à plus ou moins 0.5% de différence. Une différence plus prononcée est néanmoins remarquable sur le plan : *{Attack}*. Concernant le temps de construction d'un plan, le changement de coûts n'a aucun impact. Dans les deux cas le planificateur met un temps de l'ordre de la dizaine de micro seconde pour proposer un plan. Cette analyse est observable sur la Figure 3, qui présente les diagrammes à moustaches des temps d'exécution des plans en fonction de la configuration et de leur taille.

3 Discussion et conclusion

Dans la configuration actuelle de G.O.A.P. pour le jeu F.E.A.R., les coûts des actions sont très peu déterminants pour le contrôle en temps réel des PNJs : ils n'influent ni sur le temps de construction ni sur le comportement des PNJs. Bien que les coûts des actions soient un paramètre essentiel pour la conception du planificateur utilisant l'algorithme de recherche A*, ils ne sont pas impactant car, d'une part les plans sont courts (98.9% n'ont au plus que deux actions) donc ils ne permettent pas d'exploiter le potentiel d'A*, et d'autre part le système G.O.A.P. dans F.E.A.R. utilise deux méthodes d'élagage particulièrement efficace qui permettent de sélectionner des actions candidates précises. La première sélection se fait à l'aide des postconditions, le planificateur ne garde que les actions qui possède le bon effet. Ensuite, une méthode d'élagage utilisant la mémoire et les capteurs du PNJ est implémentée dans chaque classe d'action. Cette méthode permet de ne garder que les actions qui, parmi toutes celles qui ont le bon effet, sont acceptables dans le contexte de la situation. Ainsi, si un PNJ n'est pas au contact du joueur mais est en mesure de tirer, l'action *Attack* pourra être candidate tandis que l'action *AttackMelee* sera ignorée bien que les deux actions aient le même effet : *"TargetIsDead = true"*. Cette méthode d'élagage contextuelle supplante le rôle des coûts en réduisant le facteur de branchement de A*. Elle explique l'impact si peu important présenté Figure 1. La probabilité d'apparition plus prononcée pour le plan *{Attack}* dans la configuration CU ou la diversité de 5,14% de la configuration CO est due au peu de fois où A* a eu plusieurs propositions et où la configuration du coût en fonction de la spécificité de l'action a eu, pour CO, ou aurait eu, pour CU, un intérêt.

Références

[1] Chris Conway. Goap in tomb raider. GDC AI Summit, 2015.

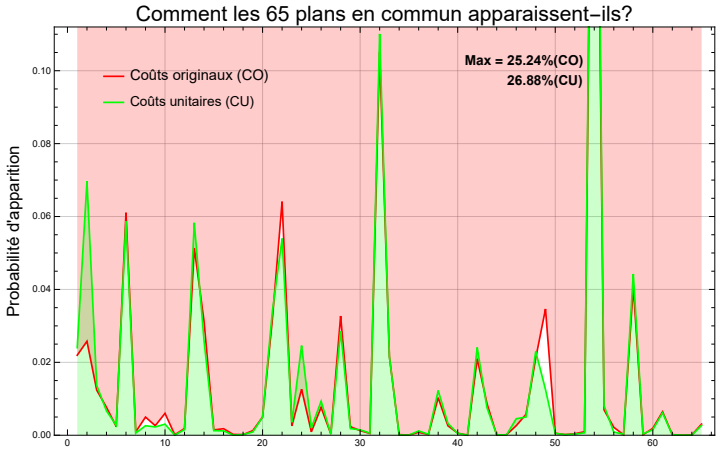
[2] Richard Fikes and Nils Nilsson. STRIPS : A new approach to the application of theorem proving to problem solving. *AI Journal 2(5)*, pages 189–208, 1971.

[3] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on System Science and Cybernetics*, pages 100–107, 1968.

[4] Peter Higley. Goal-oriented action planning : Ten years old and no fear ! GDC AI Summit, 2015.

[5] Jeff Orkin. *Applying Goal-Oriented Action Planning to Games*. 2003.

[6] Jeff Orkin. Agent architecture considerations for real-time planning in games. *Proceedings of the* 1^{st} *AIIDE*, pages 105–110, 2005.



{{1 → {Animate}, 2 → {Attack}, 3 → {AttackFromAmbush}, 4 → {AttackFromArmored},
5 → {AttackFromArmoredBounded}, 6 → {AttackFromCover}, 7 → {AttackFromVehicle},
8 → {AttackGrenadeFromCover}, 9 → {AttackLungeUncloaked}, 10 → {AttackMelee},
11 → {AttackMeleeLong}, 12 → {AttackMeleeUncloaked}, 13 → {AttackReady},
14 → {AttackTurretCeiling}, 15 → {DismountVehicle}, 16 → {DodgeCovered},
17 → {DodgeOnVehicle}, 18 → {DodgeRoll}, 19 → {DodgeRollParanoid}, 20 → {DodgeShuffle},
21 → {EscapeDanger}, 22 → {FaceNode}, 23 → {FlushOutWithGrenade}, 24 → {Follow},
25 → {GetOutOfTheWay}, 26 → {GotoNode}, 27 → {GotoNode3D}, 28 → {GotoTarget},
29 → {GotoTarget3D}, 30 → {GotoTargetLost}, 31 → {GotoValidPosition}, 32 → {Idle},
33 → {IdleTurret}, 34 → {InstantDeath}, 35 → {InstantDeathKnockedDown},
36 → {KnockDownBullet}, 37 → {KnockDownExplosive}, 38 → {LongRecoilBullet},
39 → {LongRecoilExplosive}, 40 → {LongRecoilHelmetPiercing}, 41 → {LongRecoilMelee},
42 → {LookAtDisturbance}, 43 → {ReactToDanger}, 44 → {ShortRecoilMelee},
45 → {Stunned}, 46 → {SuppressionFire}, 47 → {TraverseBlockedDoor}, 48 → {TraverseLink},
49 → {UseSmartObjectNode}, 50 → {DismountNodeUncloaked, GotoTarget},
51 → {DismountNodeUncloaked, Idle}, 52 → {GotoNode, FollowWaitAtNode},
53 → {GotoNode, MountVehicle}, 54 → {GotoNode, UseSmartObjectNode},
55 → {GotoNode3D, UseSmartObjectNode}, 56 → {GotoNodeOfType, AttackFromView},
57 → {GotoNodeOfType, LookAtDisturbanceFromView}, 58 → {GotoTarget, InspectDisturbance},
59 → {ReloadCrouch, Attack}, 60 → {GotoNode, MountNodeUncloaked, UseSmartObjectNodeMounted},
61 → {SurveyArea, GotoTarget, InspectDisturbance},
62 → {SurveyArea, GotoTarget3D, InspectDisturbance},
63 → {UseSmartObjectNode, GotoNode, UseSmartObjectNode},
64 → {DismountNodeUncloaked, GotoNode, MountNodeUncloaked, UseSmartObjectNodeMounted},
65 → {GotoNode, UseSmartObjectNode, GotoNode, UseSmartObjectNode}}}

FIGURE 1 – Comparaison des probabilités d'apparition des plans en commun entre les coûts originaux et les coûts unitaires.

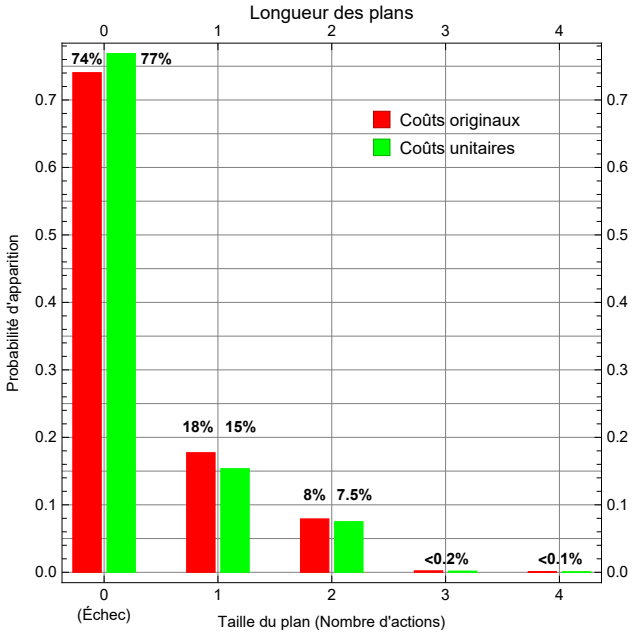


FIGURE 2 – Probabilité d'apparition des plans en fonction de leur longueur (0 étant un échec de planification).

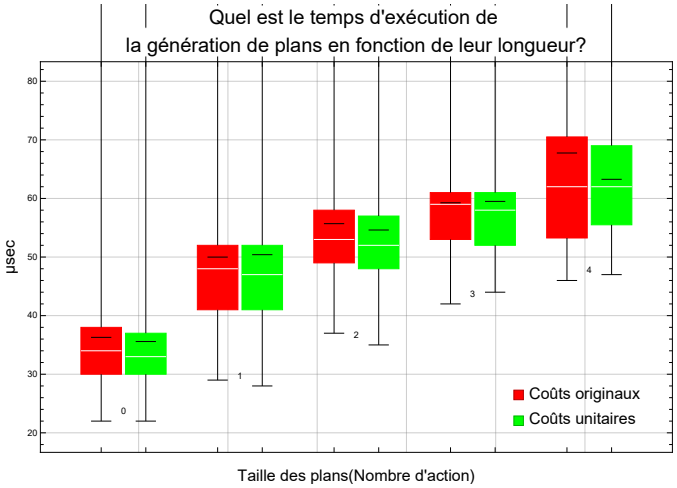


FIGURE 3 – Diagrammes à moustache des temps d'exécution du planificateur ; la marque blanche dénote la valeur médiane tandis que la marque noire dénote la valeur moyenne.