Aplicando Planejamento em Jogos do Tipo RTS

Bruno N. Luiz Carlos R. Lopes

Universidade Federal de Uberlândia, Faculdade de Computação, Brasil

Abstract

With a digital game industry in constant ascendant and a more demanding consumer, designers and programmers have faced even bigger challenges to create realistic games. This paper presents an architecture for implementation of intelligent agents and a planning algorithm for an RTS electronic game. For the definition of the planning algorithm, A and a representation system based on STRIPS were used.*

Keywords: planning, games, intelligent agents

Author's Contact:
crlopes@ufu.br
falecom@brunonepomuceno.com.br

1. Introdução

Na tentativa de se criar jogos cada vez mais reais, fabricantes de jogos frequentemente encontram desafios que são superados com teorias acadêmicas [LENT et al. 1999]. O domínio dos jogos eletrônicos, caracterizado pela sua complexidade e dinamismo, é um excelente laboratório para o teste de pesquisas, especialmente aquelas desenvolvidas na área de Inteligência Artificial [ORKIN 2004a]. Por exemplo, quando um NPC (*Non-Player Character*) precisa explorar um novo mundo, ele utiliza, frequentemente, um algoritmo A* ou de alguma de suas variações.

A indústria de jogos cresce consideravelmente a cada ano [COSTA and BELFORT 2005]. Não se tem apenas mais jogos no mercado, pode-se notar que eles são cada vez mais realistas e deparam-se com um jogador cada vez mais exigente. Com isso, as antigas técnicas para determinar o comportamento de NPCs baseadas em FSM (*Finite State Machines*) e RBS (*Rule Based Systems*) começam a ser substituídas. Ao invés de percorrer um grafo predefinido de transições, um NPC controlado por um algoritmo de planejamento procura uma seqüência de ações que satisfaça uma meta. [ORKIN 2005b]. Esta é a proposta do premiado jogo F.E.A.R da *Monolith Productions* [MONOLITH 2005].

O uso de técnicas de planejamento torna um NPC mais flexível para gerenciar situações inesperadas. Além disso, a utilização de metas e ações cria uma estrutura que facilita a manutenção, compartilhamento e reuso de código [ORKIN 2005b]. Por isso as máquinas de estado finito estão sendo substituídas por uma arquitetura GOAP (*Goal-Orient Action Planning*). Técnicas de planejamento estão intrinsecamente ligadas ao gerenciamento de recursos (escalonamento). Em um jogo do tipo RTS (*Real-Time Strategy*), normalmente jogadores devem gerenciar recursos como, por exemplo, madeira, ouro e comida. Estes recursos são necessários para realizar ataques ou se defender de ameaças [BURO and FURTAK 2003].

Entretanto, tipicamente, a obtenção de um resultado satisfatório em um processo de planejamento é marcada por problemas de desempenho. Essa situação é intensificada quando ambientes são complexos e dinâmicos como o domínio dos jogos. Desta forma, é necessária a utilização de um eficiente sistema de representação de conhecimento e um algoritmo que se benefie dessa representação para gerar planos em um tempo aceitável.

A proposta deste artigo é apresentar uma arquitetura de um agente e um algoritmo de planejamento em um jogo do tipo RTS. A área de Planejamento tem tido um desenvolvimento muito significativo nos últimos anos e muitos destes resultados poderão ser integrados ao domínio dos jogos.

2. Arquitetura do Agente

A arquitetura do agente usada no trabalho realizado foi baseada no C4 do *MIT Media Lab*. [BURKE at al. 2001]. Nesta arquitetura um agente é composto por uma unidade de memória, componentes de percepção, planejamento e execução de ações e um sistema baseado num quadro negro (blackboard) responsável pela troca de informações entre os componentes da arquitetura. A unidade de memória armazena o estado do ambiente que é relevante para a tarefa que estiver sendo executada (*Working Memory*). O componente de percepção permite a atualização dinâmica do estado do ambiente, o que se faz possível pela presença de sensores. Em função do estado do ambiente, ações podem ser executadas para que determinadas metas sejam satisfeitas. O componente de execução, também conhecido sistema motor, é responsável pela execução das ações. O componente que determina quais as ações que devem ser executadas bem, como a ordem de execução, é um sistema de planejamento.

No C4, todo conhecimento adquirido através do sub-sistema de sensores é armazenado na *Working Memory* através de objetos chamados Objetos Perceptivos de Memória (do inglês *PerceptMemory Objects*). Existem diversos tipos de conhecimentos que podem ser armazenados em Objetos Perceptivos de

Memória. A implementação realizada neste trabalho utilizou o ambiente Projeto Hoshimi, que será descrito na seção 4. Assim, são exemplos de tipos de conhecimentos armazenados: nanorobôs, posições específicas (pontos AZN e HP) e fluxos sanguíneos.

Quando, por exemplo, o sub-sistema responsável por determinar a próxima ação a ser realizada (sub-sistema Planejador) deseja se comunicar com o sistema motor para informar para qual posição o agente deve se mover, ele o faz através do *Blackboard.* A idéia é simples: o *Blackboard* possui uma série de atributos que podem receber ou fornecer valores aos diversos sub-sistemas. Desta forma, para atingir o efeito desejado, basta que o sub-sistema Planejador armazene um valor no atributo correspondente à próxima posição que o agente ocupará.

É também no *Blackboard* que são armazenadas as metas do agente. Desta forma, quando, a partir de uma percepção realizada por um sensor, o sub-sistema responsável por determinar as metas do agente determina que uma nova meta deve ser atingida, a comunicação com o sub-sistema Planejador é realizada através de um atributo do *Blackboard.*

3. Planejamento e Sistema STRIPS

Um problema de planejamento requer três entradas: uma apresentação do estado inicial do mundo, uma meta a ser atingida e uma descrição das ações que podem ser realizadas. O resultado de um processo de planejamento é uma seqüência de ações, que quando executada em um estado inicial qualquer, atingirá uma determinada meta [WELD 1999].

O planejamento como elemento da área de Inteligência Artificial surgiu das pesquisas sob busca em espaço de estados, prova de teoremas, teoria de controle e de necessidades práticas de robótica, escalonamento, entre outras [RUSSELL and NORVIG 2003]. O sistema STRIPS apresentado por Fikes e Nilsson [1971], é a primeira proposta que ilustra a integração destas influências.

STRIPS assume que os ambientes nos quais ele será aplicado são completamente observáveis, determinísticos, finitos, estáticos (mudança só acontece quando o agente age), e discretos (no tempo, ações, objetos, e efeitos). Tais ambientes não são frequentemente encontrados em situações reais. Desta forma diversas abordagens foram propostas para planejamento em que se assume que o ambiente é dinâmico, não determinísticos e parcialmente observáveis. Exemplos de algumas destas abordagens incluem Planejamento Contigente, Planejamento Reativo e Monitoramento e Replanejamento [RUSSELL and NORVIG 2003]. Contudo, pesquisas em sistemas baseados na hipótese de STRIPS continuam sendo desenvolvidas permitindo a solução de problemas complexos de forma eficiente. Exemplos de trabalhos nesta linha incluem FF [HOFFMAN and NEBEL 2001], GRAPHPLAN [BLUM and FURST 1997], BLACKBOX [KAUTZ and SELMAN 1998].

4. Sub-sistema Planejador

O sub-sistema implementado capaz de resolver problemas de planejamento foi chamado *AStarPlanner* e é baseado no trabalho de Jeff Orkin [ORKIN 2004a, ORKIN 2005b] em que utiliza o algoritmo A* [PATEL 2006] para gerar planos. Geralmente, este algoritmo é utilizado em problemas do tipo *pathfinding*, mas em essência, a tarefa é a mesma. Ambos partem de um estado inicial e através da realização de uma sequência de operações válidas, chegam a um estado final desejado.

Em um problema do tipo *pathfinding,* tipicamente, um nó do grafo gerado pelo A* corresponde a uma coordenada em um sistema 2D ou 3D. O que une um nó a outro é uma operação de movimentação. No caso de um plano que será gerado usando A*, cada nó corresponde a um estado do mundo e o que une um nó a outro é uma ação (ex.: mover para uma posição x, coletar um determinado recurso, usar um determinado recurso, atirar).

Certamente, não se pode representar todo o estado do mundo em um nó. Por outro lado, deve-se armazenar o mínimo necessário para que bons planos sejam gerados. A solução é utilizar um sistema de representação simbólica do mundo centrada no agente. Desta forma, todo conhecimento armazenado tem o agente como referência. Esse simples fato de centralização já economiza o trabalho de descobrir a qual objeto uma determinada informação se refere.

Cada conhecimento do mundo será armazenado em um objeto *WorldStateProperty* que possui: um símbolo, um tipo de dado e um valor. O símbolo representa o tipo de conhecimento armazenado. Por exemplo, posição, tipo da posição (no Projeto Hoshimi existem posições especiais como pontos AZN ou HP), quantidade de enzimas que o agente possui, etc. O tipo de dado indica se o conhecimento será representado com um valor inteiro ou uma coordenada 2D ou com outros tipos de dados. Por fim, o valor refere-se ao dado em si, 10, (20,200), etc. O sub-sistema *AStarPlanner* representa o mundo através de uma coleção de objetos *WorldStateProperty.*

O sistema de representação de uma ação se assemelha bastante ao STRIPS. Entretanto, ao invés de usar uma lista de conhecimentos que serão retirados do mundo e uma lista de conhecimentos que serão adicionados ao mundo para representar os efeitos da ação, tem-se uma coleção de objetos *WorldStateProperty.*

Suponha uma ação *Atirar* que dispara um tiro letal contra inimigo. Essa ação e o estado do mundo

(considerando apenas informações importantes para esta ação) podem ser representados por algo do tipo:

```
Estado do Mundo:
   [AgenteArmado?, InimigoMorto?]
Ação:
   [ SIM , --  ],      // Precondições
   [ --  , SIM ],      // Efeitos
```

Uma outra diferença é que, como nem todo conhecimento pode ser representado através de símbolos em uma pequena coleção, é adicionado a uma ação um método para se fazer verificação de precondições.

Suponha que em um jogo do tipo FPS (*First Person Shooter*) um NPC encontre com um personagem controlado pelo jogador. O NPC pode tentar matar o jogador com uma arma de ataque à distância ou com uma arma de contato. Normalmente, a arma de ataque a distância lhe garantirá uma posição mais confortável no que diz respeito a sua integridade física. Contudo, só poderá utilizar essa opção caso possua o tipo de munição correta para a arma de ataque a distância que possuir. Esse tipo de verificação recebe o nome de *precondição de contexto* [ORKIN 2005b]

Além desse método, uma ação possui também um outro método que será responsável por realizar modificações no mundo, no momento da execução da ação e não no momento de planejamento. Por exemplo, se a ação que deve ser executada é *MoverParaX* , um método que, efetivamente, faz com que o personagem se movimente para X deve ser executado.

Para usar o algoritmo A* é necessário estipular qual o custo das ações e qual a heurística utilizada para calcular a distância que um determinado nó (estado do mundo) se encontra da meta.

Na implementação realizada, o custo é utilizado para dar preferência por uma ação a outra. Por exemplo, a ação *AtacarArmaDistância* teria um custo menor do que *AtacarArmaContato* para que tenha a preferência de execução. Logo, cada ação possui um custo associado.

A heurística utilizada é a quantidade de símbolos não satisfeitos que um nó possui com relação à meta. Suponha que, estado inicial apresente 2 símbolos não satisfeitos com relação a meta proposta. Por exemplo,a meta de um nanorobô é possuir 20 móleculas de AZN (*NumAZNBot = 20*) e estar na posição com coordenada x igual a 100, e coordenada y igual a 200 (*Position = (100, 200)*) ao final do turno 300. Ao ser criado, esse nanorobô aparece na posição (10,100) e não possui nenhuma molécula de AZN. Uma ação do tipo *MoverParaX* reduziria a quantidade de símbolos não satisfeitos para 1 (faltaria apenas *NumAZNBot = 20*), logo o valor heurístico para essa ação seria 1.

5. Projeto Hoshimi

Conforme citado anteriormente, o ambiente usado para testar a arquitetura e o algoritmo de planejamento foi o Projeto Hoshimi [HOSHIMI 2006]. Esse ambiente foi utilizado este ano na categoria *Programming Battle* da competição mundial *Imagine Cup,* promovida pela *Microsoft Corporation*. O ambiente também foi usado na edição de 2005, mas com menos funcionalidades.

No Projeto Hoshimi, o programador tem a sua disposição uma coleção de classes para gerenciar um conjunto de nanorobôs. Esses nanorobôs são injetados, virtualmente, dentro do corpo de seres humanos (ou de outros seres vivos) com a intenção de, a priori, curar doenças. São diversos tipos de nanorobôs, cada um com uma função diferente, cada um com um conjunto de ações possíveis diferentes. O programador deve, em um determinado "mapa", fazer com que seu conjunto de nanorobôs realize um conjunto de objetivos.

Os mapas possuem pontos com características especiais. As mais importantes são os pontos AZN e HP. Um ponto do tipo AZN é um local onde um nanorobô pode coletar enzimas. Um ponto do tipo HP é um local onde um nanorobô pode depositar enzimas. Quando enzimas são depositadas em pontos do tipo HP, o time recebe uma determinada pontuação. Logo, independente dos objetivos, coletar e depositar enzimas é sempre importante.

Existe uma série de detalhes que não são relevantes para esse artigo, como a densidade das células, a presença de fluxos sangüineos ou nanorobôs com capacidade de alterar a densidade das células mas que podem ser verificados no site oficial do Projeto Hoshimi, na seção *Tutorials* [HOSHIMI 2006].

6. Trabalhos Relacionados

O trabalho desenvolvido foi baseado na proposta do pesquisador do M.I.T Jeff Orkin [ORKIN 2004a, ORKIN 2005b] em que utiliza o algoritmo A* para gerar planos e uma notação semelhante a STRIPS para representar os estados de um mundo e operadores válidos capazes de transformar um estado em outro. Em seu trabalho, Orkin realiza testes, em um jogo do tipo FPS, e sugere que a arquitetura provavelmente funcionaria em outros tipos de jogos. No trabalho desenvolvido, a arquitetura foi utilizada em um jogo do tipo RTS.

Em um outro trabalho, Hector Munoz da *Lehigh University,* utiliza técnicas de planejamento HTN (*Hierquical Task Network*) para modelar estratégias multiagentes efetivas para robôs do jogo *Unreal Tournament*(do inglês, *Unreal Tournament bots*) [MUNOZ-AVILA and FISHER 2004, MUÑOZ-AVILA at al. 2005]. Essa abordagem será mais cuidadosamente analisada pelos autores,

principalmente pois estes ainda não trataram a questão de planejamento cooperativo, que por sua vez, está presente no trabalho de Munoz.

7. Considerações Finais

A implementação da arquitetura de agente proposta bem como do algoritmo de planejamento baseado no A*, trouxe maior flexibilidade à implementação do time de nanorobôs de um dos autores usados na edição de 2006 da categoria *Programming Battle* da *Imagine Cup*.

Apesar de o time ter se classificado em primeiro lugar do Brasil, o algoritmo de planejamento era bastante limitado e baseado em máquinas de estado finito. Na estrutura antiga, para cada situação, era necessário codificar todas as possibilidades de ações.

Essa maior flexibilidade é extremamente desejável, uma vez que a SDK de desenvolvimento usada na competição é alterada todo ano. Isto quer dizer que seria necessário códificar inúmeros outros novos planos, e revisar os existentes, procurando adequá-los às novas possibilidades.

Além disso, tem-se que a idéia de utilização do A* para gerar planos em jogos do tipo FPS também funciona em jogos do tipo RTS, sem apresentar problemas de desempenho. Em seu trabalho, Orkin chega a sugerir que provavelmente seria possível, mas não explora a hipótese.

Como trabalhos futuros, os autores pretendem aplicar a mesma idéia em um planejamento multiagente, ou seja, utilização de A* e uma representação simbólica do mundo para gerar planos que serão executados de forma cooperativa por vários agentes.

Referências

BLUM, A. and FURST,M.,1997. *Fast Planning Through Planning Graph Analysis*. Artificial Intelligence, 90:281-300, 1997.

BURKE, R., ISLA, D., DOWNIE, M., IVANOV, Y., and BLUMBERG, B. 2001. *CreatureSmarts: The Art and Architecture of a Virtual Brain*. In Proceedings of the Game Developers Conference, San Jose, CA, 2001.

BURO, M. and FURTAK, T, 2003. *RTS Games as Test–Bed for Real–Time AI Research.*.Proceedings of the 7th Joint Conference on Information Science, JCIS, 2003

COSTA, S. and BELFORT, R., 2005. A indústria de Desenvolvimento de Jogos Eletrônicos no Brasil. [online]. Diponível em: [http://www.abragames.org/docs/PesquisaAbragames.pdf]

FIKES, R. E. and NILSSON, N. J., 1971. *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*. 2nd IJCAI, Londres, 1971.

HOFFMANN, J and NEBEL, B., 2001. *The FF Planning System: Fast Plan Generation Through Heuristic Search*. In: Journal of Artificial Intelligence Research, Volume 14, 253-302.

HOSHIMI, 2006. Site Oficial do Projeto Hoshimi. [online] Disponível em: http://www.hoshimichallenge.com/ [acessado em 02 de Agosto de 2006]

KAUTZ, H. and SELMAR, B., 1998. *BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving.* Working notes of the Workshop on Planning as Combinatorial Search, AIPS-98, Pittsburgh, PA, 1998.

LENT. M.V, LAIRD, J, BUCKMAN, J, HARTFORD, J., HOUCHARD, S, STEINKRAUS, K, TEDRAKE, R, 1999. *Intelligent Agents in Computer Games*, In Proceedings of The Sixteenth National Conference on Artificial Intelligence, Orlando, Fl, AAAI Press, 1999

MONOLITH, 2006. Site Oficial da Monolith Productions Inc. [online] Disponível em: http://www.lith.com/home.asp [acessado em 02 de Agosto de 2006].

MUÑOZ-AVILA, H., FISHER, T., 2004. *Strategic Planning for Unreal Tournament Bots*. AAAI Challenges in Game AI Workshop Technical Report, AAAI Press, 2004.

MUÑOZ-AVILA, H.. HOANG, H., LEE-URBAN, S.,2005. *Hierarchical Plan Representations for Encoding Strategic Game AI*. Proceedings of Artificial Intelligence and Interactive Digital Entertainment, AAAI Press, 2005

ORKIN, Jeff., 2004. *Symbolic Representation of Game World State: Toward Real-Time Planning in Games*. In: AAAI Workshop on Challenges in Game AI, San Jose, CA, 2004.

ORKIN, Jeff., 2005. *Agent Architecture Considerations for Real-Time Planning in Games*. In: AIIDE 2005 Proceedings, Marina Del Rey, CA, 2005.

PATEL, A. J, 2006. Amit's Thoughts on Path-Finding and A-Star. [online] Disponível em: [http://theory.stanford.edu/~amitp/GameProgramming/]

RUSSEL, S. J., NORVIG, P., 2002. *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice Hall, 2002

WELD, D. S. 1999. *Recent Advances in AI Planning*. AI Magazine 20(2), 93 – 123.