A evolução das técnicas de Inteligência Artificial

Bruno Duarte Correa
Thiago Dias Pastor
Department of Computer and Digital Systems Engineering, Escola Politécnica da Universidade de São Paulo, Brazil

Abstract

Since the dawn of the digital games there is a desire to replicate situations that make us somehow part of a context and closer to reality. The importance of artificial intelligence is at the convergence of this longing in simulating increasingly challenging realities. The techniques of artificial intelligence for games in general are part of a trend that argues that the role of IA is to simulate the behavior near human and not like other environments eg optimization objectives, seek to achieve levels of decision making and speed of reasoning very above an ordinary human being. Artificial intelligence in games is beginning to maximize the fun emulating a smart player just right, neither too smart nor too dumb demonstrating weaknesses purposeful. The purpose of this article is to demonstrate some techniques used in some classic and current games showing their uses and problems.

Keywords::

Author's Contact:

{bruno.duartec,thiagodiaspastor}@gmail.com

1 Introdução

Desde os primórdios dos jogos digitais existe a vontade de replicar situações que nos façam de certa forma fazer parte de um contexto.Quanto mais próximo da realidade melhor é tal experiência. A importância da inteligência artificial está na convergência desse anseio em simular realidades cada vez mais desafiadoras. As técnicas para jogos em geral fazem parte de uma vertente que defende o papel da IA como o de simular um comportamento próximo do humano, não como em outros ambientes com objetivos por exemplo de otimização buscam atingir níveis de decisão e velocidade de raciocínio muito acima de um ser humano comum. A inteligência artificial em jogos tem por princípio maximizar a diversão emulando um jogador inteligentes na medida certa, demonstrando fraquezas propositais. A proposta desse artigo é demonstrar algumas técnicas clássicas e algumas utilizadas nos jogos atuais mostrando os seus usos e problemas com o intuito de desmistificar a IA de alguns jogos e, como em um *PostMortem*, estimular o uso de tais técnicas.

2 Histórico

No princípio, a inteligência artificial nos jogos tinha o papel de alimentar máquinas caça níquel com o objetivo de manter o jogador por horas e horas entretido e gastando dinheiro. Jogos como Pong e Pac-Man utilizavam listas pré determinadas de ações e algumas poucas tomadas de decisão aleatórias na tentativa de tornar os jogos um pouco mais interessantes. Com o tempo, jogos começaram a utilizar técnicas um pouco mais avançadas, mas ainda assim incipientes. Entretanto na década de 80 e 90 houve uma grande reviravolta com jogos preocupando-se com o papel da inteligência artificial em títulos como *Age of Empires II* e *Warcraft II*. Em 1998 a Valve revoluciona com *HalfLife* e um incrível avanço nos jogos em primeira pessoa. Em 2000 jogos como *The Sims*, totalmente focados na experiência do jogador com a inteligência artificial em jogos que aprendiam com os jogadores, também contribuiram para a evolução do mercado. A grande realidade é que a inteligência artificial começou a tomar corpo nos jogos quando as empresas começaram a levá-la a sério como realmente uma área de desenvolvimento que deve entrar no processo e não encarada como um adicional no jogo que em geral poderia ser desenvolvido no último quarto de tempo do projeto, o que acabava por gerar comportamentos previsíveis e jogos nem tão desafiadores quanto poderiam ser. A visão atual da inteligência artificial é a de que ela seja totalmente direcionada a contribuir com o *gamedesign* de tal forma que facilite as modificações de acordo com a concepção do jogo, podendo traduzir sentimentos e sensações de forma a tornar a imersão dos jogos muito maior, e não mais encarar as técnicas de inteligência artificial como uma ferramenta para simplemente melhorar a experiência do usuário.

3 Motivação

A evolução das técnicas utilizadas em jogos eletrônicos é evidente nos últimos anos, tornando a experiência muito mais imersiva e trazendo jogos cada vez melhores.É de suma importância conhecer o estado da arte atual para aplicar tais recursos.

4 Técnicas básicas

A classificação quanto à técnicas clássicas e técnicas atuais aqui empregada foi feita pensando na utilização das mesmas e não por questões temporais como as mais antigas ou mais recentes.

4.1 Agentes

Agentes são entidades capazes de perceber o ambiente através de sensores e modificá-lo através de atuadores. O nível de percepção e o raio de atuação dos mesmos determina diretamente a qualidade da nossa abstração do agente. A maneira como o conhecimento de um novo acontecimento no mundo é transmitido entre outros agentes pode determinar uma reação justa ou não por parte dos agentes que não participaram de fato do evento. Um exemplo clássico desse problema é demonstrada em um jogo de tiro, quando o personagem principal é avistado por um inimigo e imediatamente todos os outros sabem sobre a sua localização sem um prévio aviso.

4.1.1 Percepção

Como explicitado no tópico anterior, a modelagem da percepção que o agentes faz do cenário dita em muito a qualidade da inteligência artificial. Uma abordagem é colocar sensores representando os cinco sentidos. Os mais comuns em ordem de usabilidade são visão e audição, mas podemos ter representações de olfato ou tomadas de sensação de temperatura dando uma maior relevância para as informações de cena, auxiliando na decisão dos agentes. Outra abordagem colocando em evidência a percepção de cena como um todo e não a percepção micro de um só agente são os *blackboards*, que tem por objetivo compartilhar informações de acontecimentos em uma base comum de tal forma que todos os agente tenham conhecimento. Essa abordagem é a mais utilizada, porém é preciso tomar cuidado para não tornar injusta a reação dos agentes.

Figure 1: *Sátira da percepção de um agente*

4.2 Maquinas de estado

O controle de eventos ou comportamentos é totalmente ligado ao contexto do jogo. Sendo assim, separar em momentos bem definidos facilita em muito a segregação de eventos, uma melhor depuração de problemas e implementação de novos comportamentos.

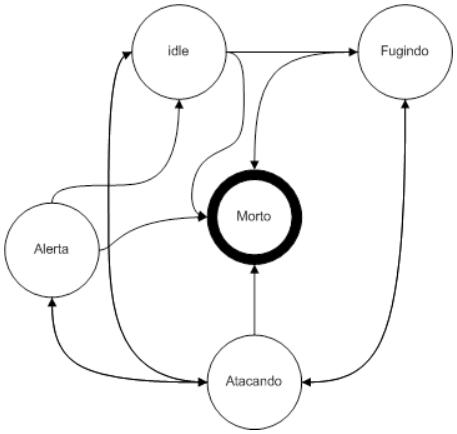

Figure 2: *Maquina de estados de um agente*

Máquinas de estado são úteis em todos os tipos de jogos por facilitarem a organização de informação em momentos e servir como controle de tomada de decisão.

4.3 Navegação

Os algoritmos de busca de caminho são a implantação de inteligência artificial mais utilizadas em jogos eletrônicos, tendo soluções das mais rudimentares às mais avançadas e em essência podem ser divididos em:

- **busca cega**: é a busca em que o agente não tem conhecimento do cenário tendo apenas como indicador do caminho a seguir os sensores e uma função objetivo que dita o quão bom é tomar o próximo passo
- **busca informada**: o agente de alguma forma tem informação do ambiente em que se encontra, podendo formular uma solução, conectando o ponto de origem ao ponto de destino de forma mais acertiva, em geral são soluções mais elegantes, nos dando a impressão de um movimento mais fluido. Devemos entretanto nos preocupar em dosar o conhecimento dos personagens para não acabarmos com a naturalidade de uma busca por caminho que um ser humano comum precisa fazer para ir de um ponto A a um ponto B.

Os algoritmos de navegação, por serem um grande objeto de estudo da robótica, são uma área muito desenvolvida e geralmente subdividida em um nível mais alto de abstração em busca de caminho *(pathfinding)* e fuga de obstáculos *(obstacle avoidance)*. Nos jogos atuais essa diferença foi desaparecendo gradativamente, visto que raramente encontramos situações em que no meio de nosso caminho não existam outros agentes ou mesmo obstáculos móveis.

4.3.1 A star

É uma busca do tipo informada, ou seja, o agente tem conhecimento do cenário em que se encontra. Tem como algoritmo geral a Busca pela Melhor Escolha *(best-first-search)*. De posse de uma arvore de possibilidades do próximo passo a tomar, ou próximo nó a escolher. A grande diferença do *A star* para as outras implementações de best-first-search é que além de levar em conta o melhor próximo nó também leva em consideração a distância já percorrida na equação, não somente o custo da expansão do nó atual.

4.3.2 *WayPoints*

Uma abordagem um pouco mais refinada para a busca de caminho é o uso de *WayPoints* [ai blog 2008], muito famosos em jogos como *Counter Strike*.

Figure 3: *WayPoints em Halaa*

Assim como as árvores de possibilidades nas busca informadas, os *WayPoints* são utilizados como modelo de representação do cenário, porém de forma mais visual e intuitiva. Os *WayPoints* são nós bases sob os quais são interpoladas as posições intermediárias. Como observado nessa imagem, em alguns casos é necessário um número muito grande de nós e caso esses não forem bem posicionados podem causar zig-zags na movimentação dos personagens. Para ambientes mais complexos não é difícil notar que essa é uma fonte de erro muito grande. É possível notar também que mesmo com um número bem grande de pontos, ainda assim existem áreas em que a qualidade da solução está totalmente ligada à maneira como interpolamos as informações, visto que elas de fato não existem o que pode ocasionar no clássico problema do personagem andando rente à paredes ou estruturas. A utilização dessa técnica não prevê qualquer informação fora do grafo, o que dificulta a retomada do caminho no advento de um obstáculo móvel ou mesmo a mudança do cenário por exemplo com a geração de um buraco durante uma batalha. Os *WayPoints* ainda são muito utilizados em jogos eletrônicos e muitas vezes se mostram suficientes para resolver o problema da navegação, mas em situações um pouco mais complexas e querendo representar a movimentação de forma mais fluida ela se torna inviável.

4.4 Algoritmos Genéticos

Algoritmos genéticos (AG) são em essência ferramentas de busca local com um toque elevado de aleatoriedade, em geral utilizado para procurar soluções fugindo do problema dos mínimos locais,

atingindo resultados inesperados, o que para o universo dos jogos eletrônicos se torna muito interessante. Assim como o processo de evolução humana, os AG passam por etapas como:

- **Herança**: durante o processo de geração da próxima geração alguns genes são herdados do ”pai” ou da ”mãe”, portanto de forma aleatória é selecionada a origem de cada um.
- **Mutação**: alguns genes após o processo de herança formando o descendente canônico são modificados para aumentar o fator aleatório da próxima geração.
- **Seleção**: segundo uma função objetivo que dita o quão bom uma solução é, alguns descendentes são eliminados ou selecionados
- **Cruzamento**: assim como no processo de mutação alguns genes foram modificados, na etapa de cruzamento de cromossomos uma fatia do mesmo é trocada com outro cromossomo

Um grande desafio de se obter AGs usuais para o universo de jogos eletrônicos é o fato que esse processo assim como o processo de evolução natural demorar para convergir para uma solução ótima, nesse caso devemos com uma função objetivo bem calibrada atingir uma solução boa. A decisão de como utilizar AG na solução do problema em geral também esbarra na definição do cromossomo. Uma boa solução é totalmente dependente de uma boa modelagem do seu cromossomo. Em geral AGs são bem empregadas para definir comportamentos de personagens obtendo a partir de um dicionário de possibilidades uma enorme variedade de personagens únicos e inusitados, sendo utilizado portanto massivamente em jogos de estratégia.

4.5 Redes Neurais

As redes neurais, como citado em [Darryl Charles1], assim como os algoritmos genéticos, tentam imitar um comportamento da natureza, nesse caso busca emular o funcionamento do nosso cérebro que como já sabemos transmite e armazena informações através de neurônios por meio de suas conexões e sinapses. A proposta das redes neurais é formar a partir de uma base de dados conhecida e um processo de treinamento uma simulação de aprendizado, modificando conexões entre neurônios e seus pesos de importância. Assim como nos algoritmos genéticos a modelagem dos neurônios é diretamente ligada à qualidade da solução e à velocidade de aprendizado. A idéia de criar um jogo que aprenda com o jogador ou mesmo treiná-lo na fase de balanceamento sempre empolgou os desenvolvedores entretanto raramente é feito de maneira aceitável. O aprendizado é basicamente dividido em:

- **Supervisionado**: fornecemos para a rede neural dados de entrada e a resposta desejada, sendo assim o sinal de entrada é propagado por toda a rede e caso atinja uma solução, tal comportamento é reforçado, do contrário as conexões e pesos são refeitas de forma a aderir á solução.
- **Não Supervisionado**: tenta a partir de uma base de dados inicial representar a estrutura estatísticas dos dados de entrada, sem saber na verdade a solução, tenta a partir dos dados interpretar o comportamento esperado.
- **Reforço**: é o comportamento que mais se assemelha à maneira como aprendemos sozinhos, por tentativas e erro avaliando se estamos mais próximos de atingir o resultado esperado. Por exemplo como aprendemos a andar, a dirigir ou a lutar.

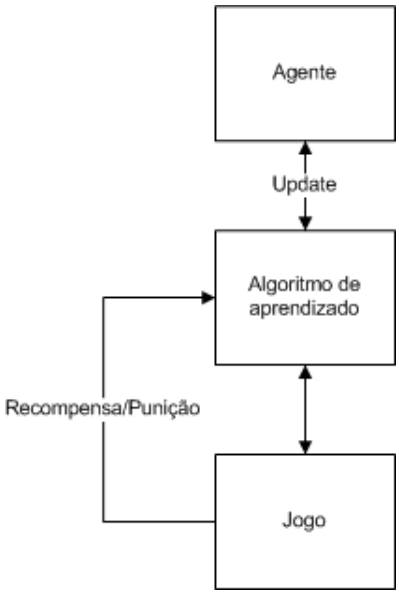

Figure 4: *Aprendizado por reforço*

O grande problema da rede neural mal desenvolvida é o aprendizado não convergir para a solução esperada podendo, no extremo, por exemplo com um aprendizado não supervisionado após varias entradas erradas, treinar a rede para responder de forma inesperada. As redes neurais podem ser empregadas em dois momentos

- **Balanceamento**: de posse de uma massa de dados de testes feitos por *testers* calibramos a rede para responder de acordo com um comportamento previamente conhecido
- **Durante o jogo**: com o uso de sensores e atuadores a rede interpreta durante o jogo os comportamento e balanceia constantemente os pesos e conexões. Essa abordagem se não bem dosada pode gerar partidas muito difíceis e desestimular o jogador, além de ser mais complexa a formulação do algoritmo de aprendizado. Jogos como Black and White usam massivamente o aprendizado por reforço durante o jogo, retirando *feedbacks* dos jogadores na tentativa de compreender os seus desejos.

Figure 5: *Cena de Black and White*

Jogos sociais como *The Sims* também utiliza aprendizado por reforço em tempo real para compreender o modo como o jogador gostaria que o seu avatar se comporte.

5 Técnicas atuais

5.1 Behavior Tree

É uma maquina de estados finitos hierárquicos disposta na forma de uma árvore. De forma mais explícita é um grafo direto aciclico e cada nó pode ter várias conexões permitindo o reuso de *sub behavior trees* como citado em [Pilosu]. As *behavior trees* vieram para substituir a intangível solução de construir um grafo como uma máquina de estados com transições e ações definidas por uma abordagem mais simples com uma passagem por uma árvore de comportamentos hierarquicamente aninhados, além de permitir a modificação em tempo real de maneira bastante intuitiva.

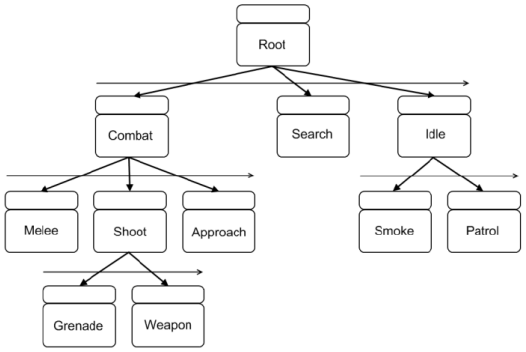

Figure 6: *Árvore de comportamento*

Inicialmente o agente deve ser modelado como uma tabela de características e flags que serão modificados e consultadas durante cada ciclo de passada pela árvore, formando a base de conhecimento do agente. Tal tabela deve ser feita com o mínimo possível de informações para facilitar a depuração. A cada ciclo, rodamos a base de conhecimento de cada agente na árvore fazendo testes condicionais com o objetivo de atingir uma folha da árvore e executar uma ação. As ações usualmente são pedaços de código que serão executados caso a folha seja atingida na passagem da árvore executando uma busca em largura. Uma característica da *behavior tree* é a fácil adição de novos comportamentos mesmo em tempo real simplesmente anexando uma folha a algum nó. Cada nó é composto basicamente por uma condição, uma lista de filhos e uma ação podendo a lista e a ação serem nulas. Podemos adicionar táticas de grupos por exemplo simplesmente adicionando ao nó uma quantidade máxima de participantes além da sua condição. Atualmente utilizado em jogos como Halo 2,Halo 3,Crysis,Left 4 Dead e vários outros títulos.

5.2 Navegação

Os jogos atuais tem abordagens com resultados mais fluidos do que os apresentados na seção anterior, conferindo ao movimento mais naturalidade e possibilitando uma melhor manipulação do mundo, facilitando a criação e modificação.

5.2.1 Steering Behavior

Steering Behaviors como citado em [Reynolds] é um tópico bastante grande e não restrito a navegação. Diversos efeitos interessantes como Flocking podem ser obtidos com o uso desta técnica. Neste paper nos restringimos a Steering Behaviors aplicado a navegação, em especial aos behaviors : *AvoidObstacle, Seek, StayOnPath* e *AvoidNeighbors*. Existem diversas maneiras de enxergar os Steering Behaviors, foi escolhida a abordagem baseada em campos elétricos pela sua fácil compreensão. Cada objeto no mundo virtual cria uma campo atrator ou repulsor. (Conceito bastante semelhante aos campos elétricos). Se um agente estiver sob o raio de ação de um destes campos, uma força propor- cional ao valor dele irá aparecer no agente. A intensidade e o decaimento destes campos são parâmetros de difícil medição cujos valores vêm da experiência do designer e da experimentação

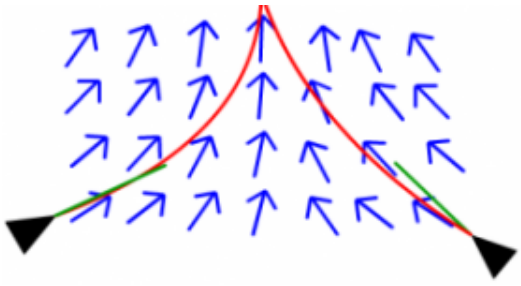

Figure 7: *Campo sobre um agente*

O projeto de um sistema de navegação seguindo *Steering Behaviors* é completamente diferente dos sistemas clássicos usando *PathFinding*. Não existe fase de pré-processamento para geração de *Way-Points* além da diferença na semântica das variáveis que neste caso estão sempre relacionadas com a física. O primeiro passo de um sistema de *Steering Behaviors* é definir um agente(o agente de *Steering* contém alguns atributos a mais do que um agente de *PathFinding*) que normalmente é modelado como um veículo (entidade que possui uma aceleração, um torque e uma velocidade máximas, além de um vetor direção que aponta para a sua frente e um sistema auxiliar que irá converter a resultante que o agente está submetido em variáveis físicas como aceleração). Este agente irá possuir alguns comportamentos chamados na técnica de *Behaviors*. Um comportamento descreve como o agente irá enxergar os campos do mundo virtual (atrator ou repulsor). A seguir os comportamentos usados pelo sistema de navegação serão descritos:

- ***AvoidObstacle***:comportamento bastante simples que consiste em enxergar os campos dos obstáculos estáticos como repulsores que decaem com a distância.
- ***Seek***: comportamento que consiste em enxergar o Destino como um atrator, ele será um campo uniforme e não decairá com a distância. O Parâmetro de entrada é a posição do destino
- ***AvoidNeighbors***: comportamento bastante complexo (sua explicação detalhada esta fora do escopo do artigo) que consiste em evitar colisões com outros agentes. O método usado é baseado em previsão futura de posição (supor que os agentes manterão a mesma aceleração e velocidade, e prever sua posição em um tempo futuro próximo, verificar se existe alguma colisão em potencial e se houver enxergar a posição da colisão como campo repulsor

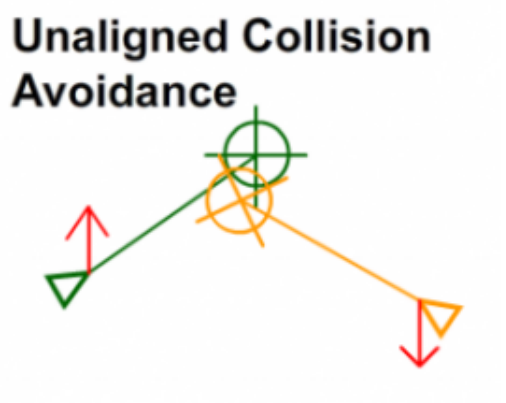

Figure 8: *Comportamento AvoidNeighbors*

- ***StayOnPath***: comportamento que mantém o agente atraído a um percurso. Fazendo uma analogia, seria como se o agente

estivesse dentro de um tubo com atração elétrica alta sendo assim o agente sempre tenderia a voltar para a sua envoltória.

Existem muito outros comportamentos para o *Steering Behavior*. A grande vantagem de se utilizar essa abordagem é obtermos comportamentos mais dinâmicos e podermos adicionar novos comportamentos a qualquer momento visto que com a abordagem de campos elétricos cada comportamento não passa de um vetor a mais somado ao movimento do agente. A utilização isolada entretanto pode causar problemas muito sérios como por exemplo o personagem ficar preso em quinas ou devido ao mal calibramento dos campos um objeto não conseguir chegar ao seu objetivo. Uma forma de corrigir tal problema é com uma abordagem híbrida com *A star* criando previamente um caminho entre origem e destino e com um comportamento *StayOnPath* garantir que o caminho traçado teria os pontos bons do *A star*, a garantia de convergência, e do *Steering Behavior*, movimentação fluida e com adição de comportamentos.

5.2.2 Navigation Mesh

Os *Navigation Mesh* como citado em [ai blog 2008] são representações do mundo assim como os *WayPoints* citados na seção anterior, entretanto ao invés de representar o modelo como uma lista de pontos, utiliza um modelo que demonstra os locais onde os agentes podem se locomover. Os *meshes* podem ser gerados em modelodores 3D em geral no mesmo momento que o designer está desenvolvendo o modelo do cenário já tem a possibilidade de desenvolver o terreno de locomoção.Como os meshes são representações de posições livres para andar enquanto não houver movimentação ou modificação do cenário os testes de colisão podem ser reduzidos consideravelmente pois garantimos que nenhum NPC vai tentar atravessar alguma parede ou colidir com algum objeto. Com o uso dos *WayPoints* tinhamos regiões que eram interpoladas, com o uso dos *Navigation Mesh* temos uma densidade de informação bem maior com uma quantidade de dados armazenado muito menor como mostra na figura

Figure 9: Navigation Mesh *em Halaa*

Como temos uma densidade de informação bem maior podemos ter uma interpolação entre pontos muito mais suave, e reações para desviar de obstáculos moveis que não tinham sido previstos no advento da criação da cena por exemplo. Ao invés de representar o mapa como um grafo de pontos conectados utilizamos como um grafo de polígonos convexos. *Navigation Mesh* são muito utilizados ultimamente por exemplo em jogos como: *Halo 2,Halo 3,First Encounter Assault Recon (F.E.A.R.),Counter-Strike: Source,Metroid Prime,Metroid Prime 2: Echoes,Metroid Prime 3: Corruption, Jak and Daxter: The Precursor Legacy,Jak II,Jak 3,Uncharted: Drake's Fortune,Scarface: The World is Yours* e muitos outros

5.3 Planning GOAP (Goal-Oriented Action Planning)

GOAP como citado em [Orkin] e [Orkin 2006] é uma técnica para tomadas de decisões que produz uma sequência de ações (plano) para atingir objetivos. O sistema é composto por:

- ***Goal***: É qualquer condição que o agente deseja satisfazer
- ***Action***:é um passo atômico no *Plan* que faz o personagem fazer algo
- ***Plan***:uma lista de *Actions*

É um processo que não elimina as máquinas de estado mas simplificam em muito a sua utilização além de tornar dinâmica a sua criação, sendo composto por um estado inicial, uma meta (*Goal*) e ações atômicas para concluir assemelhando bastante com a idéia de uma maquina de estados, entretanto ao contrário desta as conexões podem ser várias, obtendo uma grande gama de possibilidades o que aumenta em muito a diversidade das soluções, impedindo que o jogador aprenda como o jogo se comporta e burle a inteligência artificial. Essa estrutura dinâmica possibilita tanto soluções inesperadas quanto uma fácil adição de novos comportamentos.

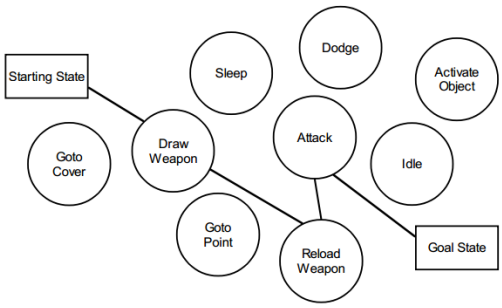

Figure 10: *Processo de formação de solução*

A criação dos planos é feita por uma busca em um grafo de ações em geral utilizando A star, considerando os nós como os estados e as arestas como ações. As vantagens sobre a solução clássica de tomada de decisão das maquinas de estado é que a princípio não é necessário determinar previamente todas as conexões que levam até as metas sendo que a única coisa que temos que especificar são pré-condições e efeitos para cada ação. Devido à flexibilidade e a facilidade de depuração e criação dos planos esta técnica, apesar de considerada nova, está tomando força frente aos novos jogos utilizada em jogos como F.E.A.R.(X360/PS3/PC), Condemned:Criminal Origins (X360/PC),S.T.A.L.K.E.R.:Shadow of Chernobyl (PC), Mushroom Men: The Spore Wars(Wii), Ghostbusters(Wii),Silent Hill:Homecoming (X360/PS3), Fallout 3 (X360/PS3/PC), Empire:Total War (PC), F.E.A.R. 2: Project Origin (X360/PS3/PC), Demigod(PC), Just Cause 2 (PC/X360,PS3), Transformers: War for Cybertron (PC/X360/PS3), Trapped Dead (PC), Deus Ex: Human Revolution (PC/X360/PS3)

6 Tendências

A evolução da inteligência artificial tem de estar sempre junto com a evolução da qualidade gráfica dos jogos, do contrário todo o esforço em computação gráfica será em vão, obtendo jogos frustrantes. A simulação de eventos ou comportamentos tem de ser cada vez mais verossímil. Por isso existem desenvolvimentos de ferramentas de teste e calibração dos algoritmos de forma mais intuitiva para que o designer possa modificar os dados sem a ajuda de um programador. Com a migração de parte do processamento da inteligência artificial para as placas de vídeo o poder computacional empregado crescerá bastante sendo possível por exemplo o desenvolvimento de simulação de multidão mais realistas e cut scenes interativas eliminando os cinematics pré renderizados. Uma forte tendência é aumentar o poder das IAs online podendo evoluir o aprendizado de redes neurais a partir de informações dos jogadores .

7 IA na GPU

Devido ao crescente papel da inteligência artificial nos jogos eletrônicos o percentual ocupado de cada ciclo de atualização começou a tornar-se significativo, influenciando no tempo de renderização, simulação física ou *gameplay*.Como não podemos reduzir a qualidade das simulações físicas ou diminuir a qualidade visual do jogo, a única solução era utilizar o tempo ocioso da placa de video enquanto não estivesse renderizando para fazer cálculos para IA. É fato que ainda são poucas as implementações que se utilizam de tal benefício, mas a tendência é a de cada vez mais migrar para a GPU, visto que a mesma possui poder de processamento muito superior ao da CPU para cálculos vetoriais ou repetitivos.

References

AI BLOG, 2008. Why waypoints aren't for pathfinding, 7.

DARRYL CHARLES1, S. M. The past, present and future of artificial neural networks in digital games.

ORKIN, J. Applying goal-oriented action planning to games.

ORKIN, J. 2006. Three states and a plan: The a.i. of f.e.a.r. *GDC*.

PILOSU, R. Coordinating agents with behaviour trees.

REYNOLDS, C. W. Steering behaviors for autonomous characters.