Universidade Federal de Santa Maria
Departamento de Eletrônica e Computação
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica Avançada
E-mail: pozzer@inf.ufsm.br
Período: 2007/2

Game AI

"AI programming is often more of an art then a science." [Lars Lidén [4]]

1. Game IA

Módulo do jogo responsável pela inteligência dos personagens autônomos ou qualquer coisa que possa ser usado para tornar personagens **aparentemente** inteligentes. A idéia de estudar Game AI é Conhecer uma grande gama de soluções para diversos problemas de modo que quando um problema surgir, pode-se escolher a melhor estratégia para a solução do problema. Esses conhecimentos também podem ser aplicados em áreas não relacionadas a jogos.

2. IA para jogos 3D de ação

Jogos de ação geralmente requerem respostas rápidas, ou seja, a AI deve responder o mais rápido possível a ataques do jogador. Além disso, nestes jogos que permitem vários jogadores simultâneos (*multi-player games*), os jogadores que geralmente estão acostumados a jogar contra outros jogadores, esperam comportamentos semelhantes aos humanos dos agentes, também chamados NPC (*Non-player characters*) ou B*ots* (*A software program that imitates the behavior of a human*).

"***The terms AI and Bots are almost one in the same***. *AI stands for 'artificial intelligence.' and generally represents all enemies within a game that are not controlled by a human player. This would include all the enemies you encounter in a game. Bots are a bit more specific, in that they are the AI that appears in multi player game types. They do everything a human player might, in regards to trying to kill you, as well as protect you if you happen to have a Bot AI on your team.* [1]" Para mais detalhes sobre exemplos de *bots*, consulte [2].

3. Arquitetura em Camadas do Game IA

O Game AI é geralmente incorporado ao projeto do jogo como um módulo separado que faz interface com o núcleo do *engine*. O módulo de IA é comumente dividido em camadas, como mostra a Figura 1 [5].

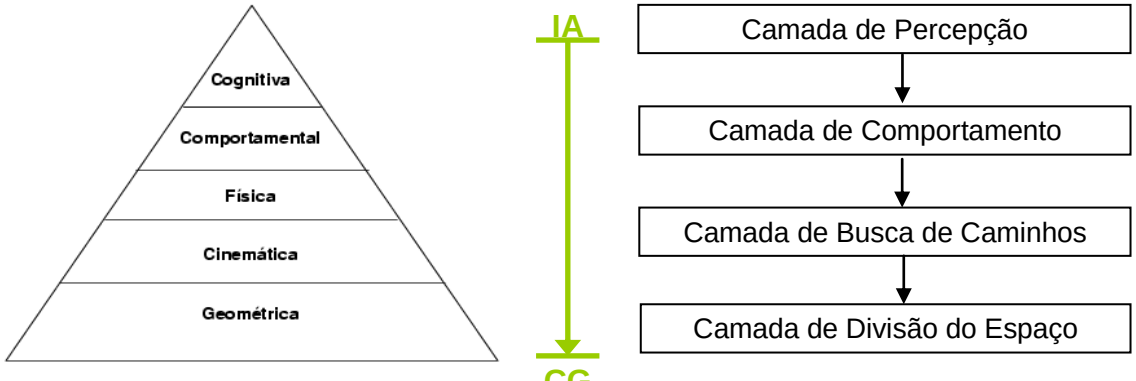

Figura 1: Camadas do Game IA

- Na camada de Subdivisão do Espaço, implementa-se a subdivisão do espaço do jogo, definindo-se unidades (nós) nos quais a AI irá operar

- Na camada de busca de caminho, implementam-se algoritmos de busca de caminho para encontrar um caminho "razoável" entre dois nós no espaço do jogo.

- Na camada de Comportamento, implementam-se algoritmos de tomadas de decisão, geralmente por meio de máquinas de estados, alem de seleção de destino para o personagem deve ir.

- Finalmente, na camada de percepção, implementam-se os recursos necessários para que o personagem possa perceber o mundo do jogo, e determinar entradas que irão alimentar a camada de comportamentos.

4. Evolução do Game IA

O campo da IA para jogos existe desde o **surgimento do vídeo game nos anos 70**. Durante muitos anos, o conceito de AI permaneceu um tanto inalterado. Porém, nos últimos anos a AI sofreu uma grande evolução. À medida que os recursos gráficos evoluíram de modo assustador, sentiu-se necessidade de fazer a IA acompanhar esta evolução. Hoje em dia, o que determina o sucesso de um jogo não é mais somente os gráficos, mas especialmente a IA, visto que é a partir dela que se impõem novos patamares de interatividade e desafios, que são a chave para chamar a atenção e despertar o desafio dos jogadores.

Entre os **primeiros jogos da história** estão Pong (primeiro Jogo de computador), Pac-man (com seus personagens *Inky*, *Pinky*, *Blink* e *Clyde*) e Space Invaders. Esses jogos faziam uso de regras simples e seqüências pré-definidas de ações, combinadas com um esquema de tomada de decisão aleatório, para tornar os comportamentos menos previsíveis.

Atualmente, um dos gêneros de jogos que fazem uso massivo da AI e que são os mais desafiador são jogos de tabuleiro, como o xadrez, visto que fazem uso de técnicas de IA tradicionais e que geralmente requerem soluções precisas em um universo de espaço de busca gigante.

Um dos gêneros pioneiros no uso da IA são jogos de estratégia. São jogos que têm como principal característica o pouco uso de recursos gráficos e um uso massivo de IA. O Game IA desse tipo de jogo é desafiador visto que requer o uso da IA a nível de unidade e a nível de jogador. No nível de unidade, são necessárias eficientes estratégias para controle de escolha de caminho para centenas de unidades simultâneas (como ocorre no WarCraft II, por exemplo). No nível de jogador, requerem estratégias táticas extremamente complexas, que sejam capazes de planejar ataques contra inimigos e ao mesmo tempo manter a defesa fortificada. Além disso, jogos deste gênero possuem vários tipos de unidades, com pontos fortes e com pontos vulneráveis às outras unidades.

Um jogo que é marco do uso da IA é o *Black & White*, da empresa Lionhead Studios. O jogo é inteiramente construído no conceito de ensino e treino das criaturas. Graças a recursos de aprendizado (*machine learning*), as criaturas podem evoluir.

Além da evolução do hardware, necessário para processar a IA a níveis considerados satisfatórios, o que contribuiu fundamentalmente para sua evolução foi que ela passou a ser considerada como fundamental para o sucesso do jogo. Ela pode não necessariamente refletir as características cognitivas humanas, mas com um pouco de criatividade, produz resultados que até o momento deixam as pessoas um tanto impressionadas com a "ilusão de inteligência" de muitos personagens.

5. Ilusão de Inteligência [3]

Para jogos que são single-players, ou para pessoas que não têm conexão de rede, uma forma de atrair a atenção do jogador é criar personagens autônomos que agem como se fossem outros jogadores dispersos em uma rede. Neste aspecto, não se espera criar unidades inteligentes, mas sim criar uma "ilusão de inteligência". ("*We want to build intelligent actors, not just intelligent thinkers*" - Martha Pollack, from Computers and Thought Lecture, IJCAI-91). Em outras palavras, esperam-se criar comportamentos que imitem comportamentos humanos.

Para dar aos *bots* comportamentos convincentes, deve-se incluir:

- Evitar roubar pela IA (*AI cheating*);

- Geração de caminhos que se pareçam naturais (evitar viradas bruscas e tratar corretamente obstáculos);
- Bons mecanismos para avaliação do cenário;
- Percepção semelhante a dos humanos.

Quanto ao aspecto de roubo pela IA, existem diversas opiniões e pontos de vista. Muitas vezes, para tornar o jogo mais dinâmico, é necessário que as unidades façam uma análise do terreno para escolha do caminho mesmo sem antes terem percorrido o terreno para saber como ele é. Em outros casos, para tornar o jogo mais desafiador, pode ser necessário dar aos NPCs capacidades "além do natural" que venham a suprir a deficiência de inteligência e improviso que os humanos possuem. Apesar da IA ser deficiente em alguns aspectos, por outro ela pode ser mais **precisa**, mais **eficiente** e mais **rápida**, o que pode ser caracterizado, pelo ponto de vista do jogador, como roubo.

6. Criando Erros Intencionalmente [4]

O que torna um jogo divertido não corresponde necessariamente à criação de NPCs mais espertos. Afinal de contas, supostamente o jogo é feito para que o jogador humano vença. Porém, a vitória não pode ser resultado de uma má implementação da IA dos personagens. Deseja-se que os erros dos oponentes sejam intencionais, ou seja, deve-se sintonizar a aplicação de modo que os erros sejam plausíveis e ao mesmo tempo não dêem ao personagem um comportamento não inteligente. Deve-se observar que os humanos são inteligentes e também erram. O programador deve saber medir onde e em qual nível a inteligência deve ser adicionada. Isso inclui possíveis roubos, desde que não sejam claramente perceptíveis pelo jogador (comportamentos não naturais). A ilusão de inteligência deve ser independente das técnicas utilizadas.

Criar um personagem que possa vencer um humano é fácil. O difícil é fazer um que perca para um humano em uma batalha desafiadora. Ou seja, o personagem deve demonstrar suas habilidades e mesmo assim deve perder. A seguir são descritas algumas técnicas utilizadas para atingir este propósito, que são principalmente direcionadas a jogos do tipo FPS.

1. **Mova antes de atirar**: Sempre que entra em um cenário novo, se gasta um certo tempo para fazer o reconhecimento do ambiente. Durante este tempo, deve-se evitar que os NPCs ataquem o jogador, especialmente em situações onde os NPCs estão escondidos. O jogador pode levar um tiro e até mesmo morrer sem saber quem o matou e de onde veio o tiro. Uma solução para isso é fazer com que pelo menos um NPC locomova-se no cenário antes de iniciar o ataque para alertar o jogador que a batalha esta por começar;
2. **Seja visível**: É meta de todo programador de IA fazer com que os NPCs permaneçam o mais escondido possível, para que possam fazer emboscadas aos jogadores. Isso ocorre especialmente em cenários com camuflagem. Para isso, deve-se usar uma textura dos NPCs que tenha um certo contraste com o ambiente do jogo;
3. **Tenha uma péssima mira**: Em jogos onde se dispõe de muita munição, deve-se tomar o cuidado para que o jogador não seja morto em uma fração de segundo. Uma solução simples é fazer com que os NPCs tenham uma porcentagem de erro dos tiros disparados Outra solução consiste reduzir o efeito de cada tiro sobre o jogador. Essa segunda solução, porém, inibe alguns benefícios secundários da primeira abordagem. Para o jogador, é extremamente contagiante poder ver balas traçantes passando próximos a sua cabeça ou batendo em paredes, produzindo pó ou movimentação de objetos. Outro fator é que os jogadores vão achar que o NPC errou o tipo devido a suas habilidades de locomoção e de se esquivar;
4. **Erre o primeiro tiro**: Outra forma de evitar matar o jogador de modo imediato, especialmente com o uso de armas de alto poder de destruição, ou em ataques pelas costas do jogador, é errar o primeiro tiro. Nenhum jogador gosta de morrer sem pelo menos saber quem o acertou. Por isso, é sugerido que o primeiro tiro erre o jogador, ou acerte em algum lugar próximo a ele, de modo que fique alertado. Outra forma é chamar a atenção do jogador por meio de algum sinal sonoro que o jogador possa escutar. Deve-se lembrar que o objetivo do NPC não é matar o jogador e sim criar tensão;
5. **Ataques individuais**: Em situações onde existem muitos oponentes simultâneos, uma solução adequada é fazer com que poucos ataquem o jogador a cada momento, ou mais especificamente, que ocorra um revezamento de quem ataca (como ocorre em filmes de artes marciais). Os NPCs que não estão atacando podem estar recarregando as armas ou procurando por kits médicos, por exemplo;
6. **Adição de vulnerabilidades**: Ao invés de fazer com que o jogador descubra vulnerabilidades dos NPCs, uma dica é incorporar explicitamente as vulnerabilidades aos NPCs. Por exemplo, um NPC que está

correndo deve demorar mais para trocar de arma e disparar do que um que está parado. Quando um NPC é atacado por traz, ele pode ficar surpreso de ter uma reação mais demorada. NPCs que sabem desviar de minas, em certas situações de perigo podem ir em direção a elas. Isso dá mais realismo e personalidade aos NPCs.

Para que todas estas dicas sejam postas em prática de modo eficiente, deve-se dedicar grande parte dos esforços da produção do jogo as fases de testes, para avaliar o *gameplay* (jogabilidade). Nesta fase, jogadores, que geralmente não fazem parte da equipe de desenvolvimento, devem testar exaustivamente cada característica para detectar possíveis imperfeições na IA (algo que caracterize erro de programação), bem como para encontrar possíveis estratégias dominantes. A fase de testes deve iniciar durante o desenvolvimento do jogo, e não somente após a sua conclusão. Os testes devem ser realizador por vários jogadores e devem existir observadores presentes em cada jogada de teste.

7. Dicas gerais de IA [3]

A seguir são apresentadas diversas dicas de implementação do Game IA que são resultado de relatos que equipes de desenvolvimento de jogos:

1. **Definição do foco da IA**: qual aspecto é mais importante a ser tratado na IA para o jogo em questão: geração de caminhos suaves, comportamentos inteligentes , aprendizado, criação de estratégias sofisticadas ou respostas rápidas a eventos do jogo?
2. **Estimativa dos recursos financeiros, humanos e tempo disponível**: devem-se implementar recursos de IA que atendam a estes dois requisitos;
3. **Mantenha simples**: KISS – *Keep it simple stupid*. Essa é a expressão que resume a abordagem a ser adotada no desenvolvimento de um jogo ou qualquer outro aplicativo. Sistemas de IA geralmente são baseados em muitos parâmetros e desvios, que rapidamente podem se tornar muito complexos e conseqüentemente, fora do controle, inclusive para depuração. Deve-se quebrar o problema complexo em partes simples, que possam ser facilmente compreendidas, depuradas e reutilizadas. Isso ocorre principalmente na implementação de Máquinas de Estados Finitos - FSM;
4. **Pré-compute a navegação:** O tratamento de locomoção de personagens em jogos é mais fácil do que a movimentação de agentes um ambiente 3D porque em um jogo pode-se roubar. Ao invés de usar cálculos complexos de geração de caminhos e testes de interseção, deve-se fazer um pré calculo dos principais caminhos possíveis e adicionar esta informação ao cenário, permitindo assim que aos agentes navegarem rapidamente e a um baixo custo computacional em qualquer parte do cenário. O pré-processamento da malha geralmente é feito pelo designer, por meio de ferramentas ou manualmente;
5. **Coloque a inteligência no mundo e não no personagem**: Ao invés de criar um modelo de AI que possa tratar todas as situações possíveis, deve-se criar uma IA simples que permita ao agente escolher, em tempo real, a ação a ser executada. Por exemplo, no jogo The sims, sempre que o personagem se seu destino, objetos ou o próprio local dão instruções ao personagem (objetos inanimados), por meio de mensagens, do que deve ser realizado, como por exemplo, quais são os passos para que o personagem coma algo que está dentro da geladeira. Isso permite que novas funcionalidades sejam incorporadas ao jogo sem a alteração da IA do personagem. Isso dá ao personagem uma ilusão de inteligência, quando na verdade ele está apenas seguindo um script de ações previamente estipulado;
6. **Dê a cada ação um *timeout***: Algo que ninguém espera ver em um jogo é um agente realizar ininterruptamente uma ação errada. Não será percebido se o personagem virar a esquerda quando na verdade deveria virar a direita, porém qualquer um vai notar se ele ficar tentando atravessar uma parede eternamente. A AI do jogo deve monitorar o sucesso da realização das ações pelos personagens, para que no caso relatado, após um certo tempo, deve designar ao personagem insistente outra ação.

8. Referências

[1] http://vgstrategies.about.com/od/strategyglossary/g/ai_bots.htm?terms=ai
[2] http://www.botspot.com/search/s-game1.htm
[3] Steve Rabin, Ed. **AI game programming Wisdom**. Charles River Media; 1 edition, 2002.
[4] Steve Rabin, Ed. **AI game programming Wisdom 2**. Charles River Media; 2 edition, 2003.
[5] Gilliard Lopes. Notas de Aula.