

UTILIZANDO LÓGICA FUZZY PARA CONTROLE DE JOGOS

Bruno Crivelari Sanches
Universidade Federal de Itajubá – bcsanches@gmail.com

Abstract – This paper demonstrates a study about using Fuzzy Logic to control game agents, Fuzzy techniques are used to control an agent movement in a virtual environment created specific for this study. Also it is demonstrated the use of Fuzzy logic in decision making, in this case, weapon selection.

Keywords – fuzzy control, game development

Resumo - Este artigo mostra um estudo realizado sobre o uso de Lógica *Fuzzy* para controle de agentes de jogos, empregando técnicas *Fuzzy* para controlar a movimentação de um agente em um ambiente virtual criado especificamente para este estudo. Também é demonstrado o uso de *Fuzzy* na tomada de decisões, no caso, seleção de armas em um jogo.

1. Introdução

Os jogos atuais utilizam as mais variadas técnicas de inteligência artificial para criar oponentes extremamente sofisticados para concorrer com os jogadores humanos, tal sofisticação é atingida com o uso de diversos algoritmos e tecnologias de inteligência artificial, sendo que no desenvolvimento dos jogos atuais não é incomum existir uma equipe inteira dedicada à criação de um sistema inteligente [MD2001].

Uma das inúmeras áreas de inteligência artificial de um jogo consiste no desenvolvimento de algoritmos que permitam que agentes tomem decisões dos mais variados tipos, como, por exemplo, qual oponente atacar, para que local se deslocar para melhor proteção, que armamento utilizar, etc [MD2002].

A forma como as decisões para essas ações são tomadas dependem especificamente de como um especialista (um jogador humano no caso) se comportaria nessas situações.

Um problema típico dos jogos de ação é o da seleção de armas. Em [MR2005] foi analisado que humanos decidem quais armas utilizar usando regras como, por exemplo:

distância para o alvo, munição disponível e consequência.

Certas armas podem ser favorecidas devido à distância com que o jogador se encontra do alvo, por exemplo, certos jogos possuem armas que são imprecisas e disparam vários projeteis e conforme os projeteis se deslocam, eles vão ficando cada vez mais afastados uns dos outros, assim apenas uma pequena quantidade atinge o alvo. Com esse tipo de armamento o ideal é que o jogador o utilize apenas a curta distância, para maximizar sua chance de acerto e pontuação.

Outro tipo de armamento são aqueles que simulam explosões, se forem usados a curta distância eles causam danos até mesmo ao jogador que realizou o disparo, sendo assim esse tipo de arma só é utilizada quando o atacante pode manter uma distância segura do seu alvo.

Visando analisar o uso de sistemas *Fuzzy* em jogos foi criado um pequeno jogo testes que é descrito nas próximas seções deste artigo, onde é explicado como foi criado este ambiente e como foi elaborado o sistema *Fuzzy* utilizado para controle de movimentação e para seleção de armas dos agentes, e por fim, os resultados obtidos.

2. Jogo Experimental

Para o estudo de lógica *Fuzzy* foi desenvolvido pelo autor um pequeno jogo, para que fosse possível simular um ambiente de jogo o mais próximo possível do real dentro do tempo disponível para o estudo.

O jogo é baseado no *Space Invaders* [3], onde uma frota de naves invade um planeta e o jogador tem como missão destruí-las.

Apenas gráficos em duas dimensões foram utilizados, os inimigos surgem da parte superior da tela e vão descendo até atravessar toda tela. A nave do jogador fica restrita a parte inferior, podendo mover-se lateralmente.

Na versão criada para este estudo os inimigos não atiram, apenas se deslocam até sair da tela e quando um inimigo consegue sair o

jogador é penalizado perdendo um pouco da sua energia. Quando a energia do jogador chega a zero, este perde o jogo e a partida se re-inicia.

O jogador tem a sua disposição três armamentos diferentes, sendo duas armas para disparos rápidos e outra arma que dispara foguetes, de forma lenta, mas que produzem grande dano ao inimigo, dano este que pode atingir naves próximas incluindo a nave do próprio jogador, exigindo aqui prudência na sua utilização, para este não ser prejudicado também.

Todas as armas possuem um número limitado de disparos e o jogador ganha novos disparos ao destruir inimigos, esse ganho é aleatório, podendo ocorrer ou não.

O jogador também pode e deve se deslocar lateralmente, assim pode se posicionar de forma a atingir os inimigos com seus disparos.

No início do jogo o jogador é confrontado com uma pequena esquadilha de naves que se aproxima e novas naves vão surgindo com o passar do tempo. As naves são criadas em um intervalo de tempo que vai se reduzindo conforme o jogo avança, de forma a aumentar a dificuldade do jogo de forma gradual.

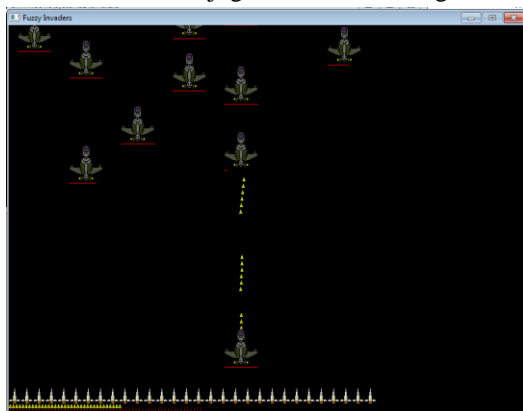


Figura 1: Jogo em pleno funcionamento

O jogo foi desenvolvido utilizando-se a linguagem C++ com o uso de algumas bibliotecas externas, incluindo *Phobos 3D[PHBS]* para processamento de entrada e gráficos, biblioteca *Fuzzy Raven[MB2005]*.

Foram feitos experimentos para utilização da biblioteca FFL [SR2002] que utiliza a linguagem FCL (*Fuzzy Control Language*), mas devido a biblioteca utilizada se integrar diretamente ao código C++ optou-se pelo seu uso para minimizar o tempo de desenvolvimento.

Para aumentar a dificuldade e deixar o experimento mais interessante a nave do jogador possui uma pequena aceleração ao se movimentar, assim, quando o jogador interrompe um comando de movimento a nave ainda se desloca por algumas unidades devido ao seu momento, isso torna o jogo mais desafiador tanto para humanos quanto para inteligência artificial.

A nave do jogador foi organizada de forma a ser controlada com o uso de controladores, sendo que cada controlador utiliza uma lógica de controle diferente, tornando simples o processo de se adicionar ao protótipo novos controles para testes de outras técnicas de inteligência artificial.

Para este estudo foram criados três controladores:

- *Humano*: controlador onde uma pessoa interagindo com o computador deve controlar a nave e atacar os inimigos.
- *Fuzzy*: este controlador utiliza, como o nome sugere, lógica *Fuzzy* para movimentar a nave até o alvo mais próximo (selecionado com o uso de heurísticas) e para decidir que armamento utilizar.
- *Random*: controlador aleatório que seleciona de forma aleatória todas as suas ações.

Cada controlador gera um comando para a nave que controla, sendo que este comando contém um número que indica a aceleração M da nave, sendo que $-1 \leq M \leq 1$, além disso é gerado um número que identifica a arma a ser usada e um valor verdadeiro ou falso indicando se a nave deve disparar ou não. Este comando é enviado à nave que se encarrega de executar as ações, assim todos os controladores controlam a nave sempre com a mesma interface, de forma que um controlador não tenha como “trapacear”.

3. Controlador *Fuzzy*

O controlador *Fuzzy* é o módulo do jogo que controla a nave do jogador com o auxílio de lógica *Fuzzy*, que é utilizada para controlar o movimento da nave e a seleção de armas, o funcionamento destes dois controles é explicado seguir.

3.1. Controlando Movimentos

Para controle dos movimentos o controlador precisa gerar valores no intervalo $-1 \leq M \leq 1$, onde -1 significa deslocar a nave para esquerda com velocidade máxima e 1 para a direita com velocidade máxima. Sendo o valor contínuo, a lógica *Fuzzy* aparece como uma boa candidata para definir o movimento que a nave deve realizar.

Primeiramente foram definidas algumas heurísticas para se decidir com qual inimigo a nave vai se alinhar ou usar como alvo. A heurística usada foi a de sempre procurar perseguir o inimigo que estiver mais próximo, essa heurística tem uma característica dar preferência a inimigos que estejam próximos da parte baixa da tela (e prestes a “fugir”).

Escolhido o inimigo “alvo”, é calculada a distância horizontal da nave do jogador em relação ao inimigo, valor este que é então usado como entrada no sistema *Fuzzy*.

O sistema possui três funções de pertinência para a variável de entrada d (distância), que possui valores válidos no intervalo de $0 \leq d \leq 800$, o seu mapeamento é mostrado na Figura 2: Regras de pertinência para distância do alvo.

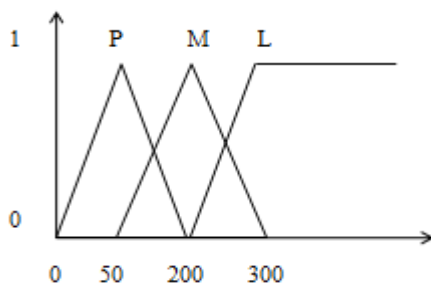


Figura 2: Regras de pertinência para distância do alvo.

O modelo lingüístico foi utilizado para se mapear a variável d , sendo assim seus valores são mapeados como P (perto), M (mediano) e L (longe).

A variável d é associada com a variável a (aceleração), que pode possuir valores no intervalo $0 \leq a \leq 1$, o seu mapeamento pode ser observado na Figura 3: regras de pertinência para aceleração..

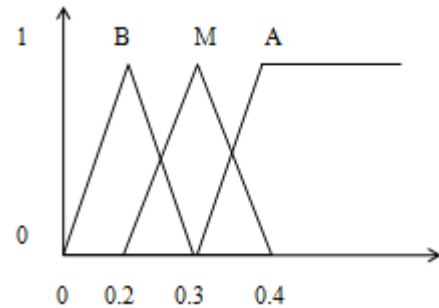


Figura 3: regras de pertinência para aceleração.

Esta variável, também foi mapeada pelo modelo lingüístico, sendo seus valores B (baixa velocidade), M (velocidade média) e A (alta velocidade).

Para associar as duas variáveis foram utilizadas três regras descritas a seguir:

- r1: IF $d = P$ THEN $a = B$;
- r2: IF $d = M$ THEN $a = M$;
- r3: IF $d = L$ THEN $a = A$;

Com essas três regras o sistema já é capaz de definir um valor de aceleração de 0 a 1 para a nave, após o cálculo do valor de aceleração pelo módulo *Fuzzy* é verificado de que lado o inimigo esta em relação a nave, caso seja do lado esquerdo, o valor é multiplicado por -1 para que nave se desloque na direção correta.

Com estas poucas regras o sistema foi capaz de movimentar a nave de forma bem precisa, fazendo com que ela se aproxime rapidamente do alvo e de forma suave.

3.2. Controle de Disparos

No controle de disparos foi utilizada apenas uma heurística que verifica se existe algum alvo na frente da nave do jogador, quando um alvo é encontrado a nave recebe o comando para realizar disparos com a arma selecionada atualmente.

3.3. Seleção de Armas

A definição da arma a ser utilizada é feita com um conjunto de regras *Fuzzy* mais complexas que as utilizadas na movimentação e foi baseado no sistema mostrado em [MB2005].

São dois critérios analisados para se decidir qual arma utilizar, sendo estes definidos como distância do alvo e munição disponível.

A distância é utilizada para se analisar qual arma pode obter maior sucesso quando o alvo esta perto ou longe, o jogo possui, por exemplo, um lançador de foguetes, que a curta distância pode causar danos ao próprio jogador, que deve evitá-la ao extremo nessas situações.

Armamento com pouca ou nenhuma munição também não é desejável, sendo assim essa variável foi modelada para se evitar o uso de armas nessas condições.

O sistema produz como resultado o nível de usabilidade de uma determinada arma, sendo este valor um percentual que pode variar de 0% a 100%, foi empregado um algoritmo para pesquisar entre todas as armas disponíveis qual possui o maior fator de usabilidade e selecionar esta para uso.

Para melhor detalhar a modelagem *Fuzzy* da seleção de armas a seguir vai ser mostrado como foi modelado o lançador de foguetes. Todas as armas possuem modelagem quase idêntica (variando apenas os valores utilizados e as condições das regras) ao lançador de foguetes, assim as outras armas vão ser omitidas deste documento.

A primeira entrada do sistema é a distância do alvo em relação ao jogador, esta distância é representada pela variável d que pode variar no intervalo $0 \leq d \leq 1000$. Esta variável é modelada com três funções, como mostrado na Figura 4: regras de pertinência para distância do alvo para seleção de armas..

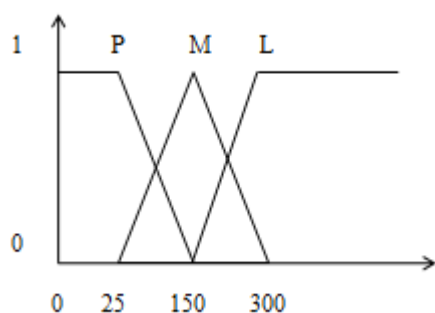


Figura 4: regras de pertinência para distância do alvo para seleção de armas.

As funções podem obter os valores P (próximo), M (mediano) e L (longe).

A variável de entrada q (referente a quantidade de munição) pode possuir valores no intervalo $0 \leq q \leq 100$ e foi modelada utilizando três funções, conforme ilustrado pela Figura 5: regras de pertinência para quantidade de munição..

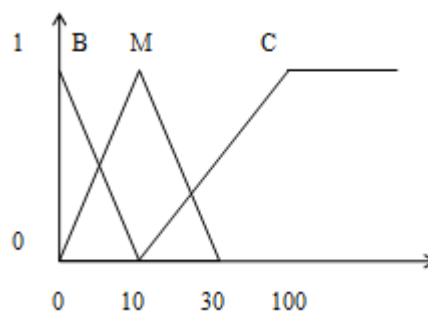


Figura 5: regras de pertinência para quantidade de munição.

Esta variável é modelada com valores B (baixo), M (mediano) e C (cheio) para indicar a quantidade de munição disponível.

A usabilidade de uma arma é modelada utilizando-se também três funções de pertinência, conforme é mostrado na Figura 6: regras de pertinência para usabilidade de uma arma.

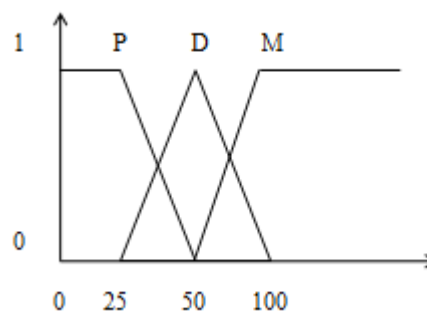


Figura 6: regras de pertinência para usabilidade de uma arma.

A modelagem lingüística foi feita utilizando-se as referências P (pouco desejável), D (desejável) e M (muito desejável).

A partir destas três variáveis foram definidas nove regras que são listadas a seguir:

- r1: IF $d = P$ AND $q = C$ THEN $a = P$;
- r2: IF $d = P$ AND $q = M$ THEN $a = P$;
- r3: IF $d = P$ AND $q = B$ THEN $a = P$;
- r4: IF $d = M$ AND $q = C$ THEN $a = M$;
- r5: IF $d = M$ AND $q = M$ THEN $a = D$;
- r6: IF $d = M$ AND $q = B$ THEN $a = D$;
- r7: IF $d = L$ AND $q = C$ THEN $a = D$;
- r8: IF $d = L$ AND $q = M$ THEN $a = P$;
- r9: IF $d = L$ AND $q = B$ THEN $a = P$;

Pelas regras podemos inferir que o lançador de foguetes é muito utilizado quando o inimigo esta a uma distância média ou longe, de

preferência quando se tem muita munição disponível.

Estas escolhas foram feitas devido ao foguete ser a arma com menor capacidade de munição do jogo, então é desejável economizar de forma a se utilizar este em situações críticas.

Outro aspecto deste armamento é que o seu projétil tem um deslocamento lento, devido a isto é preferível utilizar este armamento quando o alvo se encontra a uma distância mediana (como mostrado pelas regras r4, r5 e r6).

4. Resultados Obtidos

Durante os testes foi observado que a nave controlada pelo sistema *Fuzzy* obteve um bom resultado comparado com as outras formas de controle que foram empregadas.

Para efeito de comparação o jogo foi disponibilizado para alguns jogadores que na sua maioria obtiveram resultados inferior ao do controlador *fuzzy*, em especial quando existem muitos inimigos ativos.

Por outro lado o controlador em alguns momentos demonstra certos erros ou atitudes que não são comuns a um controlador humano, como, por exemplo, deixar naves fugirem em situações onde ele poderia facilmente ter atacado e evitado a fuga, isso ocorre devido a simplicidade das regras e da heurística usada para seleção de alvos.

O controlador *fuzzy* também obtém um ótimo desempenho para selecionar armas, sempre utilizando a arma mais eficiente sempre que possível de forma a eliminar os inimigos o mais rápido possível.

Nos testes realizados foram observados raríssimos casos onde o controlador *fuzzy* causou dano a si mesmo com o uso do lançador de foguetes.

5. Conclusão

Neste trabalho foi feito uma experimentação do uso de lógica *fuzzy* para controlar elementos de um jogo. O resultado final se mostrou bastante satisfatório e promissor.

Após modelagem das regras e algumas experimentações foi possível obter rapidamente bons resultados e os ajustes ao sistema são simples de se realizar devido ao fato de os módulos *fuzzy* concentrarem todos os valores que influenciam nos resultados, diferente de outras técnicas onde é necessário ajustar o código em diversos pontos do sistema.

Para trabalhos futuros pretende-se estudar a criação de sistemas que permitam levar em consideração mais fatores que possam causar uma boa influência nas decisões, como, por exemplo, verificar se existem aglomerados de inimigos e dar preferência a utilização a um armamento que possa causar danos a vários inimigos. Outra proposta de estudo é evitar que a nave *fuzzy* colida com as demais naves do jogo e adicionar disparos aos inimigos para aumentar a dificuldade da simulação.

6. Referências Bibliográficas

[MB2005] Buckland, M. Programming Game AI by Example. Wordware Publishing Company, Inc. 2005.

[SIWK] Space Invader. Wikipedia, http://en.wikipedia.org/wiki/Space_Invaders

[PHBS] Sanches, Bruno. Phobos 3D Game Engine. <http://code.google.com/p/phobos3d>

[SR2002] Rabin, Steve. AI Game Programming Wisdom. Charles River Media.

[MD2001] Deloura, Mark. Game Programming Gems 2. Charles River Media, 2001.

[MD2002] Deloura, Mark. Game Programming Gems 3. Charles River Media, 2002.