Estudo do Uso de Técnicas de Inteligência Artificial em Jogos 2D

Soraia Teixeira Barbosa
Universidade Severino Sombra, CECETEN, Curso de Sistemas de Informação, soraiatbarbosa@gmail.com

Janaína Veiga
Universidade Severino Sombra, CECETEN, Curso de Sistemas de Informação, janainavcarvalho@gmail.com

Carlos Vitor de Alencar Carvalho
Universidade Severino Sombra, Centro Universitário de Volta Redonda,Centro Universitário Geraldo di Biase e FAETEC-IST/ Paracambi, cvitorc@gmail.com

Resumo: *A Inteligência Artificial (IA) serve como um recurso para promover uma interação mais "convincente" nos jogos, sendo, em jogos mais complexos, um elemento chave. A IA pode ser encontrada em todos os tipos de jogos, desde os mais simples, como damas e xadrez, até os mais complexos, como RPGs online e simuladores. Com a vinda de empresas de jogos eletrônicos para o Brasil, novas oportunidades vem surgindo para os interessados no desenvolvimento de jogos e com isso o número de estudantes interessados no assunto aumentou consideravelmente. Neste artigo será apresentado um estudo sobre sistemas especialistas aplicados em jogos, tomando como exemplo um sistema de batalhas em um jogo de RPG. Todo o código desenvolvido está disponibilizado em um endereço eletrônico para servir como objeto de estudo para estudantes da área, professores e pessoas em busca de conhecimento sobre o assunto.*

Palavras-chave: *Máquina de estados finita. Sistema baseado em regras. Inteligência Artificial. Jogos.*

Study on the Use of Artificial Intelligence Techniques in a 2D Game

Abstract: *Artificial Intelligence (AI) works as a resource responsible to promote a more convincing interaction in games, as it is a key element in more complex games. The AI can be found in all types of games, from the simplest like checkers and chess to the most complex, such as role playing games and simulators. With the coming of electronic gaming companies in Brazil, new opportunities are emerging for those interested in game development and, therefore, the number of students interested in the subject has increased considerably. In this article, it will be quoted an artificial intelligence algorithm to be demonstrated through the battle simulator developed for this purpose. All code developed is made*

available in an electronic address to serve as an object of study area for students, teachers and people in search of knowledge on the subject.

Keywords: *Finite state machine. Rules-based system. Artificial Intelligence. Games.*

Introdução

A Inteligência Artificial (IA) tem forte presença na maioria dos jogos eletrônicos, permitindo que a interação seja mais dinâmica e passando ao jogador a sensação de que a máquina realmente está pensando e se comportando como um ser humano na tentativa de vencê-lo. A IA pode ser encontrada em todos os tipos de jogos, desde jogos tradicionais como damas e xadrez até jogos mais modernos, como RPGs *online* e simuladores como o famoso *The Sims*, um simulador de vida onde o jogador "vive" em um mundo virtual e interage com inúmeros personagens controlados pela máquina (Figura 1).

Figura 1. Tela do Jogo The Sims. [Fonte: <http://3.bp.blogspot.com/ > Acesso em: 28 out. 2011.]

"Jogos com jogabilidade não linear necessitam de personagens mais "espertos" e que possam usar raciocínios complexos para encontrar soluções alternativas para os problemas." (Karlsson, 2005, p.12). A complexidade do algoritmo poderá variar dependendo do tipo de interação e do nível de dificuldade do jogo.

A indústria de jogos tem apresentado crescimento no país. Segundo pesquisa realizada pela Abragames (Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos) em 2008, observou-se um crescimento significativo do setor nos últimos 2 anos e a expectativa de um crescimento ainda maior em 2009, graças à vinda de estúdios internacionais para o Brasil. A pesquisa foi focada no desenvolvimento da indústria no país, ou seja, tudo o que foi fabricado no país, seja para mercado local ou importação (Abragames, 2008, p.12-p.11).

A pesquisa realizada pela Abragames mostrou que, apesar da forte presença da pirataria no país, o Brasil consegue obter uma fatia significativa do mercado através dos ganhos com exportações. O crescimento dos jogos nas redes sociais e dispositivos móveis também tem proporcionado um aumento na indústria de jogos no país, permitindo o surgimento de pequenas empresas nesse meio. Segundo matéria publicada no UOL Economia, foi feita uma estimativa de que em 2011 os brasileiros iriam gastar cerca de R$ 3,4 bilhões em jogos online (Figura 2), colocando o Brasil como um mercado estratégico para as empresas do setor, comparável, em tamanho, aos países europeus (Uol economia, 2011).

Nos últimos anos, empresas de jogos eletrônicos como a Ubisoft[1] tem vindo para o Brasil, criando novas oportunidades para os interessados no desenvolvimento de jogos e com isso o número de estudantes interessados no assunto aumentou consideravelmente.

Figura 2. Gráfico da Estimativa de Gastos com Jogos em 2011 no Brasil [Fonte: <http://www.newzoo.com/ENG/1607-Infograph_BR_ Portugese.htm l > . Acesso em: 14 nov. 2011].

Na Internet, são disponibilizados muitos programas que auxiliam os estudantes na criação destes jogos, como por exemplo o Game Maker (GM, 2011) e o *Adventure Game Studio* (AGS, 2011). Porém, para os desenvolvedores interessados em ingressar no mercado de jogos eletrônicos, é importante conhecer uma linguagem de programação como C, C++ ou Java, pois estas linguagens são as mais utilizadas no desenvolvimento de jogos eletrônicos. A utilização de uma linguagem de programação permite desde a criação de uma *engine* própria até o desenvolvimento com o uso de *engines* e bibliotecas específicas para o desenvolvimento de jogos.

O desenvolvimento de um jogo não envolve apenas programação, mas várias áreas como *design*, música, *marketing*, entre outras e a IA é um dos principais elementos na grande maioria dos jogos. Muitos desenvolvedores iniciantes podem acabar implementando IA em seus jogos, mesmo sem saber, pois é através dessa que os personagens controlados pela máquina agem no jogo.

Atualmente, na Internet, encontramos vários materiais sobre IA, porém poucas são desenvolvidas para um público iniciante. Esta foi uma das motivações para a elaboração desse artigo.

Outra motivação para este trabalho é o fato de quase não se encontrar material na Internet com comparação e demonstração de algoritmos de IA, aplicados em jogos. A visualização do comportamento do algoritmo em um jogo pode demonstrar de forma mais clara seu funcionamento e facilitar na comparação de suas características.

Neste trabalho, o objetivo foi o desenvolvimento de um Sistema de Batalhas e, através desse, fazer um estudo sobre a utilização do algoritmo Sistema Baseado em Regras. O projeto foi todo desenvolvido utilizando *software* livre e o código-fonte está disponibilizado na Internet para servir como objeto de estudo para interessados em Inteligência Artificial aplicada em jogos eletrônicos (Barbosa, 2011).

Todas as versões do Sistema de Batalhas estão e disponibilizadas no endereço https://sites.google.com/site/sistemadebatalhasia/. No endereço eletrônico, os estudantes poderão baixar os sistemas, encontrar informações referentes ao trabalho e entrar em contato com a autora, através de um formulário de contato disponibilizado no mesmo.

Assim, a segunda seção desse artigo apresenta o conceito e técnicas de IA e sua importância na Computação e nos Jogos Eletrônicos. Na terceira seção são apresentadas a *interface* e suas principais funções do Sistemas de Batalhas. O Sistema de Batalhas tem a versão Base contendo apenas a interface e a interação do jogador e um versão feita para o algoritmo de IA tendo apenas a função de ação do inimigo alterada. Na quarta seção descreve-se o algoritmo abordado e o resultado. Finalmente, na quinta seção são apresentadas as impressões finais sobre o desenvolvimento da monografia.

Inteligência Artificial

De acordo com Valle (2011), a Inteligência Computacional (IC) é uma área da Computação Natural que estuda a natureza do homem de resolver problemas com objetivo de desenvolver um sistema computacional capaz de resolver problemas complexos, sendo essa uma área emergente de pesquisa que abrange redes neurais artificiais, teoria dos conjuntos nebulosos computação evolutiva e Inteligência Artificial.

Segundo Bittencourt (2006, p.20), o objetivo da Inteligência Artificial é simultaneamente teórico – a criação de teorias e modelos para a capacidade cognitiva – e prático – a implementação de sistemas computacionais baseados nesses modelos. A Inteligência Artificial faz com que a máquina tenha a capacidade de criar estratégias, aprender, reconhecer padrões ou encontrar as melhores soluções possíveis, simulando a forma de raciocínio de um ser humano, porém este raciocínio é implementado em um computador.

De acordo com Rich (1998, p.6), a técnica de IA é um método de explorar o conhecimento que deve ser representado de modo tal, que possa:capturar generalizações, ou seja, não é necessário que ele represente separadamente cada situação individual. Uma técnica de IA deve ter a capacidade de compreender o problema e ser usada em várias situações sem precisar de mudanças na sua estrutura. Uma técnica de IA auxilia na resolução de

problemas complexos usando conhecimento, simulando o raciocínio.

Tendo a sua IA implementada, a máquina passará a raciocinar de forma autônoma, o que descarta a necessidade de comparação do seu comportamento com o de um ser humano. Os testes geralmente são baseados apenas na interação do programa com o seu mundo. Em grande parte dos problemas, a máquina é capaz de encontrar a melhor solução sem cometer erros, o que auxilia e agiliza a realização de diversas tarefas.

A IA está presente em muitos sistemas de computação, auxiliando na realização de inúmeras tarefas, sejam elas simples ou complexas. Podemos citar como exemplo de sistemas computacionais que utilizam IA: sites de busca, programas que resolvem problemas de matemática, sites que planejam viagens na internet, programas que traçam rotas como o *Google Maps*, sistemas de reconhecimento de padrões, jogos, entre muitos outros.

Inteligência Artificial nos Jogos Eletrônicos

A IA e os jogos eletrônicos sempre foram relacionados. De acordo com Rich (1988, p. 1), alguns dos primeiros problemas da IA a serem estudados foram problemas de jogos e provas de teoremas.

> A IA e os jogos eletrônicos sempre foram relacionados. De acordo com Rich (1988, p. 1), alguns dos primeiros problemas da IA a serem estudados foram problemas de jogos e provas de teoremas. No início dos anos 60, Arthur Samuel obteve êxito na criação do primeiro programa operacional significativo de jogos. O programa jogava damas, e além de simples jogar ele aprendia com seus próprios erros e melhorava seu desempenho.(Rehm et al., 2007, p. 1)

No caso dos jogos, a IA nem sempre é utilizada com o intuito direto de encontrar a melhor solução para vencer o jogador, pois, nesse caso, a máquina pode se tornar invencível. Nos jogos o importante é dar a impressão de que a máquina está realmente competindo com o jogador, podendo haver uma margem de erros para que a jogabilidade seja mais realista.

A IA também pode ser muito útil na área de jogos educativos, fazendo com que o jogo vá percebendo as dificuldades do usuário e focando os problemas nessas dificuldades.

O Sistema de Batalhas

Para demonstração dos algoritmos propostos no trabalho foi desenvolvido um Sistema de Batalhas composto por três personagens que serão controlados pelo usuário e um inimigo que usará algoritmos de IA para tomar suas decisões. Cada personagem, inclusive o inimigo, possui os atributos HP (*Hit Points*) , MP (*Magic Points*), Força, Destreza e Agilidade.

Os personagens e inimigos foram criados a partir de estruturas (Quadro 1) e, posteriormente, foram atribuídos diferentes valores a cada um de seus atributos.

Os atributos força, destreza e agilidade foram utilizados para o cálculo do dano durante a batalha, os HP são os pontos de vida dos personagens, a batalha termina quando o HP do inimigo ou dos personagens controlados pelo jogador acaba. Os MP são os pontos de magia usados para cura, ou seja, recuperação dos pontos de HP.

Foi escolhido o desenvolvimento deste sistema com interface gráfica com o intuito de tornar a demonstração do desempenho dos algoritmos mais didática.

Quadro 1. Struct dos Personagens

```
struct personagens
{
    const char *nome;
    int hp_atual;
    int hp_max;
    int mp_atual;
    int mp_max;
    int forca;
    int destreza;
    int agilidade;
    int id;
} personagem1, personagem2, personagem3;
```

Desenvolvimento do Sistema de Batalhas

Para o desenvolvimento do Sistema de Batalhas, foi utilizado o Ambiente de Desenvolvimento *Code::Blocks* na versão 8.02. O sistema foi desenvolvido em linguagem C e para a criação da interface multimídia foi utilizada *Simple Directmedia Layer* (SDL).

A SDL é uma biblioteca multimídia escrita em linguagem C que auxilia no acesso a funções de vídeo, teclado, mouse, *joystick* e áudio. Além de auxiliar nessas funções, a SDL também apresenta a vantagem de ser multiplataforma. "A SDL suporta *Linux*, *Windows*, *Windows CE*, *BeOS*, *MacOS*, *Mac OS X*, *FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, e QNX." (*Sdl, 2011).

O inimigo e os painéis da tela foram desenhados no editor de imagens vetoriais *Inkscape* e editados no GIMP, um programa de manipulação de imagens. As edições feitas nas figuras foram basicamente na cor de fundo, que foi mudada para R: 236 G: 5 B: 236, cor escolhida para representar a transparência no sistema (Figura 3).

Figura 3. Imagem do Slime com a Cor de Fundo para Transparência

Para o plano de fundo e os sons foram utilizados recursos livres da internet. O jogo conta com uma estrutura principal que inicializa os recursos e dá início ao *loop* principal (Quadro 2) que é executado até o encerramento do jogo. No início do turno, é verificado se houve derrota, ou seja, se alguma das condições para derrota foi cumprida. Em seguida é executada a mudança de turno. Cada personagem tem direito à uma ação por turno.

Quadro 2. Loop principal do jogo

```
while (quit==false)
{
    derrota = verifica_derrota();
    if(derrota == 0){
    muda_turno();
    espera_comando(true);
    }
    desenha_interface();

}
```

A função *espera_comando* espera pelos comandos de entrada do jogador e só após a execução das ações o loop principal é executado novamente.

Interface

A interface do Sistema de Batalhas é dividida em quatro partes (Figura 4).

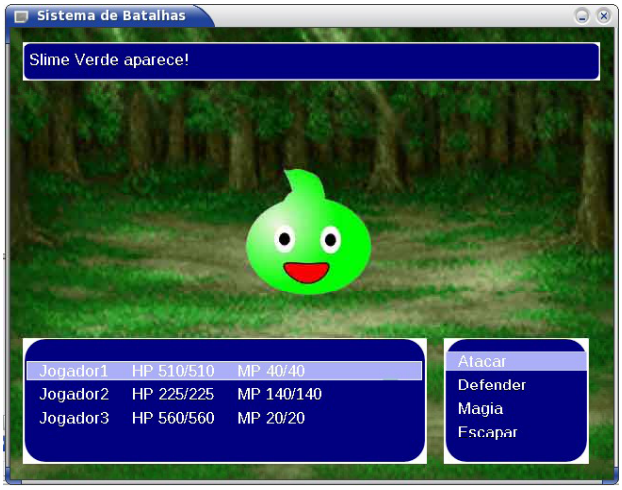

Figura 4. Interface do Sistema de Batalhas

No painel superior é exibida a última ação ocorrida no jogo. No caso da imagem acima, a batalha acabou de começar, por isso o jogo anuncia que ocorreu a aparição do inimigo *Slime*. No centro da tela está o inimigo.

No painel inferior esquerdo algumas informações sobre os personagens controlados pelo jogador. Na primeira coluna está o nome de cada jogador, na segunda seus *Hit Points* e na terceira os *Magic Points*. Os pontos de vida do inimigo não são exibidos na tela, apenas o dano causado a eles será exibido no painel superior cada vez que um ataque for realizado. No painel inferior direito está o menu com as ações do jogador. Cada personagem, seja do jogador ou da máquina, tem direito a uma ação por turno. O personagem poderá atacar, defender, usar magia ou fugir.

Ações dos Personagens

Caso o personagem escolha *Atacar* será calculado o dano no oponente escolhido baseado nos seus atributos e nos do oponente. Como pode ser visto da equação 1, o ataque é calculado através da fórmula:

$$Dn = F * (De / Ag) \qquad \text{eq.1}$$

Onde *Dn* é o dano, *F* é a Força do personagem, *De* é a destreza do personagem e *Ag* é a agilidade do oponente.

Se a escolha do personagem for *Defender*, ele ficará em posição de defesa durante todo o turno corrente, voltando à posição normal apenas quando chegar sua vez de executar um ação novamente. Quando o personagem está na posição de defesa os danos causados a ele são reduzidos, pois seu foco está na defesa, o que, no caso deste sistema de batalhas, multiplica sua agilidade por 1,5 (Quadro 3).

Quadro 3. Função de Defesa do Personagem Controlado pelo Jogador

```
void personagem_defender()
{
    //Aumenta em 1,5 a agilidade do personagem
    char_selecionado->agilidade = char_selecionado->agilidade * 1,5;
    // Seta mensagem de defesa
    sprintf(texto_painel, "%s defende.",char_selecionado->nome);
    acao = 1;
    seletorpy = 345; // Posiciona cursor para o próximo turno
}
```

Ao escolher a opção *Magia* o personagem poderá usar magia para recuperar seus pontos de vida. Cada uso de magia gasta 20 MP e recupera 200 de HP (Quadro 4). Conforme pode ser visto na figura 8, primeiro é verificado se o personagem possui a quantidade de MP necessária para executar a magia. Caso personagem não possua os pontos de magia necessários, a magia irá falhar. Se a magia funcionar será recuperado o de HP do personagem selecionado.

Por último, a opção Escapar (Quadro 5) que permite que o personagem tente escapar da batalha. As chances de um personagem escapar são baixas, por isso essa opção só deve ser usada em últimos casos. Caso a fuga seja bem sucedida a batalha termina sem haver um vencedor. Em um jogo de RPG tradicional o grupo pode sofrer consequências nesta ação como perda de dinheiro (ou moedas) ou de experiência, porém, neste sistema este não será o caso.

Quadro 4. Função de Magia do Personagem Controlado pelo Jogador

```
void personagem_magia()
{
    if (char_selecionado->mp_atual>= 20) // Se o MP é suficiente
    {
        //Aumenta 200 de HP do personagem selecionado
        char_selecionado->hp_atual = char_selecionado->hp_atual+200;
        //Subtrai 20 de MP do personagem selecionado
        char_selecionado->mp_atual = char_selecionado->mp_atual - 20;
        // Seta mensagem de cura
        sprintf(texto_painel, "%s usa magia. 200 de HP recuperado.",char_selecionado->nome);
        Mix_PlayChannel( -1, som_cura, 0); // Executa som de cura
        desenha_interface();
        SDL_Delay(1000);
    }
    else
        // Seta mensagem
        sprintf(texto_painel, "%s usa magia. Magia falhou, MP insuficiente.",char_selecionado->nome);
    acao = 1;
    seletorpy = 305; // Posiciona cursor para o próximo turno
}
```

Quadro 5. Função de Escapar do Personagem Controlado pelo Jogador

```
void personagem_fuga()
{
    int chance;
    chance = rand()% 20+1;

    if(chance == 13){
    fuga = 1;
    }else{
    sprintf(texto_painel, "Fuga falhou.");
    acao = 1;
    seletorpy = 345;
    }
}
```

O inimigo poderá executar as mesmas ações que o jogador (atacar, defender, magia e escapar). As ações destes serão decididas de acordo com o algoritmo escolhido no teste.

Na função de ataque do inimigo (Quadro 6) a fórmula utilizada para o cálculo do ataque foi a mesma do personagem controlado pelo jogador (equação 1), porém o inimigo precisa verificar se o personagem ainda está vivo para atacá-lo. Caso o personagem já tenha sido derrotado, ou seja, está com o HP iguala zero, ele passa para o próximo personagem vivo.

A função de defesa do inimigo (Quadro 7) não possui muita diferença em relação à do jogador. A defesa do inimigo é multiplicada por dois durante todo o turno, voltando ao normal em sua vez de atacar novamente.

Nos jogos em geral, quando um personagem é derrotado seu HP é sempre zero, nunca um número negativo, por isso, após um personagem ser atacado, se o seu HP assumir um valor abaixo de zero, ele recebe o valor zero no HP. Assim como na do jogador, a função de magia do inimigo (Quadro 8) recupera duzentos pontos do seu HP.

Assim como o jogador, o inimigo também tem uma em vinte chances de fuga (Quadro 9). O jogo acaba quando os pontos de vida de todos os personagens do jogador ou da máquina acabarem. O grupo que tiver todos os seus personagens com os pontos de vida zerados perde o jogo.

Quadro 6. Função Ataque do inimigo

```
void inimigo_ataca()
{
    int dano; // Variável para cálculo de dano

    Mix_PlayChannel( -1, som_ataca, 0); // Executa som de ataque

    if (personagem1.hp_atual > 0)
    {
        dano = inimigo1.forca*(inimigo1.destreza/ personagem1.agilidade);
        personagem1.hp_atual = personagem1.hp_atual - dano; //Subtrai dano do HP personagem

        if (personagem1.hp_atual < 0) //Passa HP para zero, caso se torne um número negativo
            personagem1.hp_atual = 0;
        // Seta mensagem de dano
        sprintf(texto_painel, "Slime Verde ataca Jogador1. %d de dano causado.", dano);

    }
    else if (personagem2.hp_atual > 0)
    {
        dano = inimigo1.forca*(inimigo1.destreza/ personagem2.agilidade);
        personagem2.hp_atual = personagem2.hp_atual - dano; //Subtrai dano do HP personagem

        if (personagem2.hp_atual < 0) //Passa HP para zero, caso se torne um número negativo
            personagem2.hp_atual = 0;
        // Seta mensagem de dano
        sprintf(texto_painel, "Slime Verde ataca Jogador2. %d de dano causado.", dano);

    }
    else if (personagem3.hp_atual > 0)
    {
        dano = inimigo1.forca*(inimigo1.destreza/ personagem3.agilidade);
        personagem3.hp_atual = personagem3.hp_atual - dano; //Subtrai dano do HP personagem

        if (personagem3.hp_atual < 0)//Passa HP para zero, caso se torne um número negativo
            personagem3.hp_atual = 0;
        // Seta mensagem de dano
        sprintf(texto_painel, "Slime Verde ataca Jogador3. %d de dano causado.", dano);
    }
}
```

Quadro 07. Função inimigo_defende

```
void inimigo_defende()
{
    //Aumenta em 1,5 a agilidade do inimigo
    inimigo1.agilidade = inimigo1.agilidade * 2;
    // Seta mensagem de defesa
    sprintf(texto_painel, "Slime defende.");
}
```

Quadro 08. Função Magia do inimigo

```
void inimigo_magia()
{

    if (inimigo1.mp_atual>= 20) // Se o MP é suficiente
    {
        //Aumenta 200 de HP
        inimigo1.hp_atual = inimigo1.hp_atual+200;
        //Subtrai 20 de MP
        inimigo1.mp_atual = inimigo1.mp_atual - 20;
        // Seta mensagem de cura
        sprintf(texto_painel, "Slime usa magia. 200 de HP recuperado.");
        Mix_PlayChannel( -1, som_cura, 0); // Executa som de cura
        desenha_interface();
        SDL_Delay(1000);
    }
    else
        // Seta mensagem
        sprintf(texto_painel, "Slime usa magia. Magia falhou, MP insuficiente.");

}
```

Quadro 9. Função Fuga do inimigo

```
int inimigo_fuga()
{
    int chance;
    chance = rand()% 20+1;

    if (chance == 13)
    {
        fuga = 1;
        return 1;
    }
    else
    {
        sprintf(texto_painel, "Fuga falhou.");
        return 0;
    }
}
```

Sistema Baseado em Regras

Segundo Karlsson (2005), em um Sistema Baseado em Regras, o conhecimento é definido através de um conjunto de parâmetros (variáveis) e um conjunto de regras que trabalham sobre esses parâmetros, de modo que durante a "tomada de decisão" essas regras são então processadas.

No caso de um Sistema Baseado em Regras ou Sistema Especialista, dada uma série de condições, o personagem poderá se comportar de acordo com as condições apresentadas a ele, ativando as regras e passando a agir de acordo com as ações especificadas, caso a regra seja ativada.

Diferente da máquina de estados, onde são verificados apenas os estados e as possibilidades de estados para o qual o inimigo pode passar, no sistema baseado em regras, ele irá se basear em uma série de regras e condições específicas, o que torna sua ação mais eficiente. Em um sistema baseado em regras é possível modelar um comportamento mais complexo para o inimigo, o que o torna mais próximo do raciocínio de um ser humano. "Um Sistema Baseado em Regras deve ter os seguintes elementos: Memória de Trabalho, Regras e Interpretador." (Fujita, 2005, p. 28 - p. 29).

"A memória de trabalho guarda os fatos já sabidos e também as asserções feitas pela aplicações das regras." (Fujita, 2005, p.28). Através da memória de trabalho o algoritmo irá "deduzir" os fatos sobre o jogador.

As regras são definidas a partir de uma série de condições "Se-Então" que operam com base nos fatos gravados na memória de trabalho. "Essas regras fazem com que seja disparada a função a que ela se relaciona em "Então" ou faz a mudança de estado semelhante à uma máquina de estados finita (Fujita, 2005, p.28 - p.29). As regras fazem com que a máquina analise a situação e tire suas conclusões sobre a melhor ação à ser tomada. O interpretador é a parte do sistema responsável por fazer a aplicação das regras com base no estado atual, fazendo isso de duas maneiras. Através do *forward* e do *backward chaining*.

O Sistema *Forward Chaining*

"O sistema *forward chaining* é o mais comum dos sistemas de inferência. Nele, o sistema inicia com um conjunto de fatos, e a partir destes, aplica as regras repetidamente até que o resultado desejado seja alcançado." (Fujita, 2005, p.29).

De acordo com Fujita (2005, p. 29), primeiramente ele procura identificar as regras que se aplicam ao estado atual e depois ele parte para a resolução de conflitos, escolhendo uma regra a partir de um dos três critérios: escolher a primeira regra aplicável, escolher uma das regras, aleatoriamente, e atribuir pesos às regras, escolhendo a de maior peso. Tais pesos podem ser atualizados de acordo com a frequência com que cada regra é escolhida.

O Sistema *Backward Chaining*

O *backward chaining* é basicamente o contrário do *forward chaining*. A partir de um objetivo, o sistema verifica quais regras poderiam ser aplicadas para descobrir como alcançá-lo. A partir de uma informação específica ele pode presumir o que a gerou. Por exemplo, se em um determinado jogo, para construir casas o jogador precisa de cimento e tijolos, ao notar que o jogador está construindo casas, presume-se que o jogador tenha cimento e tijolos. "O *backward chaining* é dito ser um algoritmo dirigido ao objetivo. Devido à sua natureza recursiva, o *backward chaining* não é amplamente utilizado,por questões computacionais." (Fujita, 2005, p.30).

Aplicação do Algoritmo no Sistema de Batalhas

Tanto o *forward chaining* quanto o *backward chaining* pode ser usados no desenvolvimento de jogos. No Sistema de Batalhas foi utilizado *forward chaining* com o intuito de fazer com que o inimigo tenha atitudes mais randômicas (Quadro 10).

O inimigo possui três regras definidas na própria programação. Para a escolha da ação à ser tomada diante da regra, uma função *rand()* foi utilizada.A função *rand()* gera um número inteiro pseudo-randômico dentro do escopo passado a ela. Na primeira execução da segunda regra, por exemplo é gerado um número entre 0 e 3 para que o inimigo escolha radomicamente uma das quatro ações disponíveis.

Já na terceira ele escolhe um número randômico entre 0 e 2 e executa uma das três ações disponíveis. No último if-then, não é usada a opção magia, pois o inimigo verifica que seu MP já está esgotado. Com o sistema baseado em regras as ações do inimigo se tornaram menos repetitivas e menos previsíveis, gerando um desafio maior para o jogador.

Quadro 10. Algoritmo do Sistema Baseado em Regras do Inimigo

```
// Normaliza seu status no caso de uso de defesa no turno anterior
normaliza_status_inimigo();

int acao = 0;

        if (inimigo1.hp_atual > 1000){

            inimigo_ataca();
        }

        if ( inimigo1.hp_atual < 1000 && (inimigo1.mp_atual >= 20))
        {
            //Escolhe um ação randômica
                acao = rand() % 3;

                if( acao == 0){
                    inimigo_ataca();
                }
                if( acao == 1){
                    inimigo_defende();
                }
                if( acao == 2){
                    inimigo_magia();
                }
                if( acao == 3){
                    inimigo_fuga();
                }
        }

        if ((inimigo1.hp_atual < 1000) && (inimigo1.mp_atual < 20))
        {
                //Escolhe um ação randômica
                acao = rand() % 2;

                if( acao == 0){
                    inimigo_ataca();
                }
                if( acao == 1){
                    inimigo_defende();
                }
                if( acao == 2){
                    inimigo_fuga();
                }

        }
```

Considerações Finais

Um sistema de batalhas de RPG é um sistema complexo que pode chegar a resultados variados dependendo das ações dos inimigos e do jogador. Para manter a diversão presente no jogo é importante que as ações dos inimigos pareçam ser tomadas de forma consciente, com o objetivo de vencer o jogo, criando um desafio para o jogador.

O algoritmo testado pode ser utilizado em inimigos em gêneros de jogos variados, não apenas RPG. Espera-se que este trabalho seja útil como referência a estudantes interessados no desenvolvimento de jogos, servindo como auxílio no conhecimento sobre IA, uma área muito importante nos jogos atuais. No caso do Sistema de Batalhas, o algoritmo, Sistema Baseado em Regras, apresentou um desempenho satisfatório, pois este permitiu um planejamento mais complexo para as ações do inimigo.

A IA é uma área que tem evoluído muito no desenvolvimento de jogos, pois hoje os jogos tentam estar cada vez mais próximos da realidade, porém as técnicas de IA mais simples não podem ser descartadas, pois podem servir como base para o desenvolvimento de novas técnicas.

Referências

Abragames (2011). A indústria brasileira de jogos eletrônicos: Um mapeamento do crescimento do setor nos últimos 4 anos. Disponível em: <www.abragames.org/docs/Abragames-Pesquisa2008.pdf>. Acesso em: 14 out. 2011.

AGS (2011). Ferramenta Adventure Game Studio. Disponível em: http://www.adventuregamestudio.co.uk/. Acessado em: 23 abril 2011.

Barbosa, S. T. (2011). Estudo do uso de Técnicas de inteligência artificial em um jogo 2D. 2011. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Severino Sombra.

Barbosa, S. T. (2011). Jogo Sistema de Batalhas. Acesso em : 17 de out. de 2011.

Bittencourt, G. (2001). Inteligência Artificial – Ferramentas e Teorias. Editora da UFSC. 2ª. Edição. Florianópolis, 362p.

Fujita, E. (2005). Algoritmos de IA para Jogos. 2005. 66 f. Universidade Estadual de Londrina.

GM (2011). Ferramenta Game Maker. Disponível em : http://www.yoyogames.com/. Acessado em: 23 abril 2011.

Karlsson, B. F. F. (2005). Um Middleware de Inteligência Artificial para Jogos Digitais. 2005. 126f. Dissertação (Mestrado pelo Programa de Pós Graduação em Informática do Departamento de Informática da Puc-Rio) – Puc-Rio. Setembro de 2005.

Rehm, F. et. al. (2007). Aplicações da Inteligência Artificial em Jogos.

Rich, E. (1998). Inteligência Artificial. São Paulo: McGraw-Hill.

Sdl (2001). Disponível em: <http://www.libsdl.org/>. Acesso em: 15 Outubro 2011.

Uol Economia (2011). Brasileiros vão gastar R$ 3,4 bilhões em jogos on-line em 2011, diz pesquisa. Disponível em: <http://economia.uol.com.br/ultimasnoticias/redacao/2011/09/13/brasileiros-vao-gastar-r-34-bilhoes-em-jogos-on-line-em-2011.jhtm>. Acesso em: 03 out. 2011.

Valle, M. E. (2011). Mini-curso: Tópicos de Inteligência Computacional. Disponível em:<http://www.uel.br/pessoal/valle/PDFfiles/resumo_semat08.pdf > Acesso em: 28 nov 2011.

Wikipedia. Ubisoft. (2011). Disponível em: <http://pt.m.wikipedia.org/wiki/Ubisoft>. Acesso em: 23 abril 2011.