Aplicação de Inteligência Artificial em um Simulador de Evolução

Daniel M. G. Clua [1] Vivian D. Betoni [2] Roosevelt A. Silva [3] Flávio S. C. Silva [4]

[1, 2, 3] Universidade do Vale do Paraíba – Faculdade de Ciência da Computação
Av. Shishima Hifumi, 291, Urbanova – S. José dos Campos (SP) Brasil – 12244-000

[4] Universidade de São Paulo – Depto. de Ciência da Computação
Rua do Matão, 1010 – São Paulo (SP) Brasil – 05504-090

Resumo

No presente trabalho apresentamos um jogo de simulação de vida e evolução de "seres vivos" utilizando técnicas de Inteligência Artificial, em que a forma como o usuário controla um determinado ser vivo, num mundo com diferentes tipos de ambientes, determina a evolução desse ser vivo e também das demais espécies, tendo como objetivo final atingir um nível de evolução de uma espécie dominante. O controle do personagem do jogo é parcial, visto que ele desenvolve conhecimentos e habilidades no decorrer de sua vida e pode utilizá-las conforme suas necessidades, sem a necessidade da interação com o usuário. O personagem tem atributos que simulam características biológicas como fome, sede e cansaço. Esses atributos influenciam na decisão de suas ações. Além do ser vivo dirigido pelo jogador, existem outros seres vivos diferentes, com comportamentos próprios e que têm suas evoluções determinadas por técnicas de Algoritmos Genéticos, levando em consideração o ambiente onde se encontram. Este tipo de técnica de algoritmo evolutivo é inspirado na seleção natural e acha soluções baseadas em heurísticas para problemas de otimização e busca.

Palavras-chave: Inteligência Artificial, Jogos Educacionais, Algoritmos Genéticos, Programação Evolutiva, Simulação.

Contato dos Autores:
[1] daniel.moises.gc@gmail.com
[2] vdb29586@yahoo.com.br
[3] roossilva@univap.br
[4] fcs@ime.usp.br

1. Introdução

O desenvolvimento de jogos tem tido um crescimento considerável nos últimos tempos. Com o aumento de pesquisas e o grande desenvolvimento de novas técnicas computacionais (sobretudo nos campos voltados a visualização e sintetização em tempo real) a área de desenvolvimento de jogos tornou-se, nos últimos anos, uma das áreas mais ativas no vasto campo da computação, configurando um setor muito lucrativo para a indústria de informática.

A Inteligência Artificial (IA) se apresenta como um grande aliado neste crescimento. A utilização de técnicas desta área vem ganhando importância a cada dia, sobretudo pela habilidade em resolver problemas complexos e pela necessidade de reproduzir em simulações comportamentos naturais que seguem determinados padrões. Dentre as diversas técnicas envolvidas em IA, os Algoritmos Genéticos (AG) permitem resolver de forma eficiente problemas de otimização com alta complexidade computacional. Eles se inspiram em teorias de evolução de seres vivos – notadamente a teoria Darwiniana [Silveira and Barone] – mas nem sempre permitem garantir que se encontre uma solução ótima para um problema. Esses algoritmos se enquadram em uma ampla categoria de técnicas para resolução de problemas, denominadas meta-heurísticas [Sucupira 2007].

2. Trabalhos Relacionados

Embora já existam diversos jogos de simulação que consideram este tipo de abordagem com inspiração em processos biológicos, (como por exemplo o *SimLife* [Electronic Arts 1992] e o *Spore* [Electronic Arts 2007]), esses jogos enfatizam a criação livre de seres sintéticos. Diferentemente do que encontramos nos jogos existentes, propomos no presente trabalho a elaboração de um jogo onde, com a utilização de técnicas de Inteligência Artificial, seja possível simular uma variação hereditária em um organismo geneticamente puro, ou seja, simular um ser vivo e suas possíveis alterações físicas surgidas devido ao ambiente em que vive e aos outros seres vivos que interagem no mesmo habitat natural.

3. Metodologia

Considerando que o processo evolutivo real é muito complexo para ser adaptado num modelo e que demandaria elevado custo de CPU para um jogo, procurou-se considerar somente alguns fatores mais relevantes envolvidos na idéia de evolução e que pudessem facilmente, de forma educativa, ser absorvidos pelos usuários. A idéia é que o jogo possa atingir um nível onde o usuário seja capaz de perceber, naturalmente, como ocorre o processo evolucionário e como nossas ações podem determinar nosso futuro.

O projeto – ainda em desenvolvimento – está sendo construído utilizando a linguagem C#, a

ferramenta gratuita Microsoft Visual C# Express 2005 [Microsoft 2005], bibliotecas do *XNA Game Studio Express 1.0* [Microsoft 2007] e visualização de gráficos com *Direct X*.

A seguir, são apresentados cada um dos componentes funcionais que constituem a arquitetura do sistema proposto.

3.1 Criação do Mapa

A primeira etapa no desenvolvimento do trabalho consistiu em criar o mapa do mundo onde os seres vivos do jogo habitam, com diferentes tipos de terreno: água, grama, areia, etc.

Isto foi feito criando-se uma matriz de *m* x *n*, onde cada elemento é uma instância da classe *Quadro* e representa um tipo de terreno. Isto é feito através de atributos como: *imagem* (arquivo utilizado para exibição do terreno), *posicao* (posição do quadro em relação à tela), *tamanho* (largura e altura do quadro), *numTexInf* (número indicando o tipo da textura inferior) e *numTexSup* (número indicando o tipo da textura superior). Cada quadro armazena tipos de duas texturas, inferior e superior, porque nas intersecções de diferentes terrenos, para evitar uma mudança muito brusca, é feita uma "mescla" das texturas, colocando uma em cima e outra em baixo.

Com o armazenamento das informações do mapa desta maneira, pode-se saber a qualquer momento sobre qual tipo de terreno os seres vivos estão e, baseado nisto, realizar as ações necessárias.

Foi criado um editor gráfico que apresenta uma interface de fácil utilização para o desenho de mapas. Depois de desenhado, este pode ser salvo em arquivos de extensão "*.map*", que serão depois utilizados no jogo, para criar o terreno. A Figura 1 mostra o editor de mapas:

Figura 1 – Editor de Mapas

3.2 Os Animais

Os principais componentes do jogo são os animais, os seres vivos que habitam o ambiente virtual e que vão sofrendo evoluções. Todos os seres vivos apresentam as mesmas características, mas enquanto que no ser vivo controlado pelo jogador as condições para as evoluções são verificadas constantemente e atribuídas assim que a condição é atingida, as dos demais animais são determinadas no momento de sua criação e mantidas fixas. As duas maneiras de atribuição de evoluções serão mais explicadas adiante.

Para melhor aproveitamento de memória, os animais são criados aleatoriamente em locais próximos ao do jogador e, quando estes se afastam muito, são removidos do jogo.

Um conjunto de variáveis de grande importância, armazenado na classe dos animais, é a dos atributos. Nela estão inclusos: energia, fome, sede, força, defesa, velocidade, inteligência, peso e agressividade. Estes atributos servem para indicar o estado do animal (com fome, fraco, etc.), calcular a predominância entre diferentes seres vivos, indicar a execução de ações e para determinar a facilidade que o animal terá em atingir as evoluções.

3.3 Movimentação e Passagem de Ações

A movimentação do animal controlado pelo jogador é feita, primeiramente, seguindo as ações que este indica. A ação mais básica é a de mudança de posição, onde o jogador apenas aponta um lugar na tela e o animal se movimenta em direção àquele local. Antes de fazer cada movimentação, entretanto, verifica-se se o animal pode, de fato, ocupar aquele espaço. Por exemplo, se o jogador indica ao animal para que se translade a um ambiente terrestre, mas o animal é aquático, este se movimentará até a margem e ali terminará seu percurso, não indo até o ponto selecionado pelo jogador. Ainda é possível que o jogador force a movimentação do animal, pegando-o com o cursor e soltando-o em um lugar desejado. Contudo, se o animal não tiver condições de sobreviver no lugar onde foi deixado (ser aquático colocado em ambiente terrestre, por exemplo), este perderá gradativamente sua energia e morrerá se esta for esgotada.

Além da movimentação, existem também ações específicas de interação com objetos ou outros animais. Estas ações são exibidas em um pequeno menu quando o jogador seleciona um destes objetos ou animais, podendo então selecionar a que deseja que o animal execute. Uma planta, por exemplo, pode apresentar as ações de "Comer" e "Evitar", como mostra a Figura 2. Estes objetos são descritos em classes específicas, onde são armazenadas as ações existentes para aquele objeto, a imagem que será exibida, sua posição, etc. As ações, por sua vez, também são armazenadas em classes, onde são descritos todos os seus procedimentos de execução. A classe da ação "Comer", por exemplo, descreve que, quando esta é executada, o animal deve se movimentar até a posição da planta escolhida, remover aquela instância do objeto "Planta" e incrementar os atributos do animal relativos à fome e energia.

Figura 2 – Instância de um objeto e suas ações

A princípio, o ser vivo controlado pelo jogador apenas executa ações indicadas. Entretanto, apresenta a capacidade de "aprender" a realizar determinadas ações no momento oportuno. Por exemplo, no estágio inicial, o animal não sabe que, ao sentir fome, deve se alimentar de plantas, mas depois que o jogador indica essa ação um determinado número de vezes, o animal aprende a se alimentar sozinho, quando sente fome.

Isto é obtido criando-se listas de ações. Existe uma lista para cada momento em que uma ação possa ser necessária, como por exemplo, ao sentir fome, ser perseguido, etc. Estas listas são inicialmente vazias e, à medida que o animal aprende as ações, estas são inseridas na lista que convenha. No exemplo de comer a planta citado anteriormente, a ação seria adicionada à lista do que fazer quando o animal sente fome. Quando existem várias ações numa mesma lista, estas são colocadas em ordem de prioridade. O jogo, então, verifica constantemente os atributos do animal e, se certa condição é atingida, uma determinada lista de ações é chamada.

3.4 Evoluções

Como já foi explicado, existem dois métodos diferentes para atribuir evoluções. Um para o animal controlado pelo jogador e outro para os demais seres vivos.

No caso do animal sob o qual o jogador tem controle, é verificado constantemente se foram atingidos os requisitos para cada evolução possível. Por exemplo, para que o ser vivo desenvolva barbatanas, é necessário que ele percorra uma distância x. A cada movimento, então, é incrementado um valor a um contador e quando este é maior ou igual a x, o animal recebe as barbatanas. Entretanto, os requisitos devem variar, dependendo do animal. Para que isto aconteça, o valor de incremento do contador para a evolução é calculado por uma função, na qual são usados os valores dos atributos do animal, cada um sendo multiplicado por um peso, que representa sua importância para o cálculo. Atributos com peso igual a 0 são desconsiderados, já que a multiplicação também resultará em zero. Atributos com peso igual a 1 são os mais importantes, pois a multiplicação dará como resultado o próprio valor do atributo. Estes pesos variam para cada evolução, representando as características importantes ou não para que esta seja atingida.

```
public static double
FuncaoAtributos(double[] atributos,
double multi, double[] pesos){
  double total = 0;
  for (int i = 0; i < atributos.Length;
i++){
    total += (atributos[i] * multi *
pesos[i]);}
return total;
}
```

O método *FuncaoAtributos* acima é a utilizada para calcular o valor do incremento, como explicado. Este método recebe, como parâmetros, um vetor contendo os valores dos atributos do animal, um número multiplicador (valor utilizado para escala do resultado – é multiplicado por todos os atributos) e um segundo vetor, contendo os pesos de cada atributo. Na sua execução, a variável *total* é iniciada com o valor 0 e, para cada atributo, é somada com o resultado da multiplicação do valor do atributo, do multiplicador e do peso correspondente.

```
public bool Verifica(){
  if (!atingido){
    distancia += incremento;
    return (distancia >= total);}
  return false;
}
```

O método *Verifica* acima é um exemplo do código usado para verificar se uma evolução foi atingida. Primeiro é verificado se aquela evolução ainda é possível, ou seja, se ainda não foi atingida (indicado pela variável *atingido*). Neste caso, é somado o valor de *incremento* (calculado como descrito anteriormente) ao contador (*distancia*, no exemplo). Após a soma, é retornado se o contador atingiu ou ultrapassou o total, indicando que a evolução foi atingida.

Já para os outros seres vivos, a determinação das evoluções que estes apresentarão é feita de outra maneira. No momento que um destes animais é criado, verificam-se entre todas as evoluções possíveis quais têm maior chance de existirem naquele ser vivo, considerando seus atributos e o ambiente onde se encontra. Ou seja, quais evoluções lhe garantem uma maior chance de sobrevivência.

As determinantes da probabilidade variam de acordo com o tipo de evolução. Após o cálculo, obtém-se uma porcentagem, que representa a chance que o animal tem de apresentar a evolução. É então gerado um número randômico de 0 a 100 e, se esse número for menor que o valor da porcentagem, a evolução é atribuída.

```
public bool DeterminaEvolucao(){
  int porcentagem = 0;
  if (animal.especie ==
Animal.Especies.aquatico){
     porcentagem = 70;}
  else if (animal.especie ==
Animal.Especies.anfibio){
     porcentagem = 50;}
  Random rand = new Random();
```

```
  return (int)(rand.NextDouble() * 100) <
porcentagem;}
```

O código acima é um exemplo do método de determinação da probabilidade de uma evolução onde a porcentagem é 70% no caso de um animal aquático, 50% em um animal anfíbio e 0% em qualquer outro tipo de espécie. Nas últimas linhas, é criado o número randômico e retornado se é menor que a porcentagem obtida.

Para evitar a existência de muita diferença entre os níveis evolutivos dos animais, foi feito com que o número de evoluções concedidas aos seres vivos se mantenha próximo ao das evoluções apresentadas pelo animal controlado pelo jogador.

É nesta etapa de criação de seres vivos que entra a principal utilização de Algoritmo Genético. As suas principais características são: possibilidade de manter uma população de soluções para um determinado problema; possuir um processo de seleção de indivíduos da população, baseado no grau de adaptação de cada indivíduo (*fitness*) e a existência de operadores genéticos, tais como mutação e cruzamento (*crossover*) [Silveira and Barone].

Embora existam diversos tipos de abordagens para esta técnica de IA, estas idéias básicas de Algoritmos Genéticos foram utilizadas para o desenvolvimento de métodos próprios que melhor se encaixam nas necessidades do jogo aqui descrito.

Na criação de um ser vivo, inicia-se o processamento deste a partir de uma base comum aos demais. Em seguida, de acordo com o tempo decorrido, aplicam-se diversos tipos de evoluções possíveis a este animal, seja por um resultado de reprodução ou mutação. Destas, as que mais o auxiliam a viver naquele determinado ambiente são mantidas, resultando então em um ser vivo que melhor se adapta às circunstâncias em que se encontra.

A Figura 3 mostra um exemplo de processo evolutivo de um animal, passando da espécie aquática para um anfíbio.

Figura 3 – Evolução de um ser vivo

4. Resultados

Já encontra-se desenvolvido um jogo plenamente funcional, embora ainda bastante simples. O jogo permite que a pessoa controle um ser vivo a princípio pouco evoluído, indicando ações para que este execute e adquira novas habilidades via processo evolucionário. Isto permitiu que os animais, tanto o personagem como os demais, sofressem evolução conforme a necessidade e de acordo com suas capacidades.

5. Conclusão

Os resultados preliminares apresentados têm permitido avaliar as necessidades, ferramentas e o que se pode esperar do produto final deste jogo de vida e evolução.

Adicionalmente, pretende-se considerar neste jogo um número bem maior de seres vivos num mesmo ambiente. Isto aumentará a complexidade para a sobrevivência assim como o número de soluções possíveis para o personagem escolher e adaptar-se devidamente, isto é, considerar maior diversidade como ocorre no mundo real.

Referências

SILVEIRA, S. R. AND BARONE, D. A. C., 1998. *Jogos Educativos Computadorizados utilizando a Abordagem de Algoritmos Genéticos* [online]. Disponível em: lsm.dei.uc.pt/ribie/docfiles/txt200342421140151.PDF [Acesso em: 16.Abr.2007].

SUCUPIRA, I. R., 2007. *Um Estudo Empírico de Hiper-heurísticas.* Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo.

SIMLIFE - MAXIS SOFTWARE (ELECTRONIC ARTS, INC). Disponível em: www.maxis.com

SPORE - MAXIS SOFTWARE (ELECTRONIC ARTS, INC). Disponível em: www.spore.com

MICROSOFT VISUAL C# EXPRESS 2005, 2005. Disponível em: msdn.microsoft.com/vstudio/express/visualcsharp

XNA GAME STUDIO EXPRESS 1.0, 2007. Disponível em: msdn2.microsoft.com/en-us/xna/aa937795.aspx