Proposta de um Motor de Inteligência Artificial para Jogos Digitais

Evandro Grezeli de Barros Neves [1,2]
joão ricardo bittencourt [1,2]
Fernado santos osório [1]

[1]Universidade do Vale do Rio dos Sinos – UNISINOS
Av. Unisinos 950 Cx. Postal 275 – 93022-000 – São Leopoldo - RS
{evandro.grezeli@bol.com.br} {fosorio@unisinos.br}

[2]Ludens Artis
Agência Barnabé – Cx. Postal 2001 – 94001-970 – Gravataí – RS
{jrbitt@ludensartis.com.br}

Resumo

Com o desenvolvimento da tecnologia mundial, os criadores brasileiros de jogos encontram dificuldades para poder competir e interagir com o mercado nacional e mundial. O objetivo deste artigo é apresentar uma proposta de um motor de Inteligência Artificial voltada para jogos digitais multiplataforma, visando deste modo criar uma ferramenta para simplificar e acelerar o desenvolvimento de jogos.

Palavras Chave: *Inteligência Artificial, Engenharia de Software, Jogos , Frameworks*

1. Introdução

A história de desenvolvimento de jogos digitais no Brasil é muito recente, tendo a sua primeira empresa oficialmente registrada no ano de 1992. O mercado brasileiro de jogos somente em meados do ano 2000 que teve um grande crescimento de empresas, mas mesmo assim o número de empresas hoje ainda é relativamente baixo. Oficialmente são 55 empresas registradas na Associação Brasileira de Games (ABRAGames) e a grande maioria delas ainda não tem domínio tecnológico suficiente para poder ingressar no mercado a nível mundial.

Ciente das dificuldades que os desenvolvedores de jogos digitais enfrentam, este artigo irá apresentar uma proposta de uma arquitetura computacional de um motor de Inteligência Artificial, baseando-se em técnicas atuais de Engenharia de Software e técnicas de Inteligência Artificial, onde as mesmas foram pesquisadas principalmente na série *Game Programming Gems* [1]. Para isto este artigo está organizado em três seções. Na seção 2 será feita uma breve conceituação dos principais tópicos que embasarão a estrutura do *framework*, citando suas características e as técnicas que estão sendo estudadas para a sua implementação, onde também são citados alguns outros *softwares* voltados para criação de sistemas inteligentes*;* na seção 3 será apresentado como será a estrutura deste *framework*, de que forma ele irá ser implementado; e por último na seção 4 serão feitas as considerações finais.

2. Contextualização

Com o intuito de tornar mais fácil o desenvolvimento de comportamentos inteligentes para personagens em jogos digitais, o *framework* proposto poderá tanto ser utilizado como programa isolado ou poderá ser acoplado a algum projeto de motor de jogo.

Baseando-se nestes princípios, o objetivo do presente trabalho é projetar e desenvolver um motor de Inteligência Artificial multiplataforma que possa ser integrado facilmente em jogos digitais, além de outras aplicações, tais como a simulação de robótica autônoma e outros sistemas inteligentes (agentes) que requerem algum comportamento inteligente. Para isto buscou-se na Engenharia de Software subsidio teórico que permitisse desenvolver tal arquitetura.

Investigou-se principalmente *frameworks* e padrões de projetos (*design patterns*).

Um *framework* tem uma representação física em termos de classes, métodos e objeto e é uma técnica de reuso orientado a objetos voltado para aspectos de projeto, tornando assim a arquitetura única para cada determinado jogo digital [2]. As principais vantagens do uso de *frameworks* incluem o aumento do reuso e diminuição do tempo para produzir uma aplicação. Um bom *framework* pode reduzir o custo de desenvolvimento de uma aplicação, pois permite a reutilização do seu código [2]. Já os padrões de projetos (*Design Patterns*) aumentam a segurança e a confiabilidade ao desenvolvedor, pois ele saberá que o código implementado já foi estruturado por especialistas em Engenharia de Software, e os mesmos já foram testados e utilizados em diversos programas [3]. Estes padrões se dividem por afinidades: padrões de criação, padrões de estrutura e padrões de comportamento[4]. Alguns destes padrões foram estudados com mais detalhamento, pois servirão de fundamentação para arquitetura do motor de IA. Os principais padrões [3] que foram estudados tratam-se do *Singleton, Bridge, Façade, Flyweight, Command, Iterator, State, Abstract Factory* e *Adapter*.

Para o desenvolvimento desta proposta, além do estudo das técnicas de Engenharia de Software foram estudadas técnicas de Inteligência Artificial usadas freqüentemente nos jogos digitais [1], tais como o algoritmo A*, que define as trajetórias para um agente estabelecendo pontos onde é possível sua movimentação; *Finite State Machine* (FSM), uma máquina de estados que considera entradas de modo a mudar de estado e gerar novas ações; e as diversas técnicas utilizadas em jogos do tipo *Real Time Strategy* (RTS), tais como, *line of sight, fog of war, game trees, radius of sight*.. Também estão sendo estudadas técnicas conexionistas e lógica difusa [5].

Também foram estudadas algumas tecnologias existentes que possuem objetivos semelhantes a proposta que está sendo investigada. A OpenAI [6] é uma proposta de formar um padrão e depois criar as ferramentas usando este mesmo. Possui integração com diferentes *softwares* e para a sua configuração e comunicação podem ser utilizadas as linguagens estruturais XML e CORBA. As implementações podem ser tanto com a linguagem Java ou C++ e já oferece alguns algoritmos de IA, tais como Redes Neurais Artificiais, Algoritmos Genéticos, FSM e agente móveis. O Weka [7] trata-se de uma coleção de algoritmos implementados em Java voltado para *Data Mining*; o DirectIA [8] é uma aplicação volltada para a criação de vida e comportamento artificiais, adotando um sistema baseado em agentes. Todas ações dos agentes são tomadas dependendo do ambiente, da missão, dos outros agentes e de seu estado interno. Tais agentes são gerados através de uma linguagem de *scripts*.

3. A estrutura do Framework

Após os estudos teóricos referentes aos *frameworks* e *design patterns*; análise dos problemas e técnicas de Inteligência Artificial existentes em jogos digitais; e estudo das ferramentas existentes estabeleceu-se um conjunto de requisitos que deveriam estar presente no motor de IA.

Os requisitos especificados são: (a) *Padronização:* criação de uma arquitetura que utilize um conjunto de componentes padronizados e ofereça condições para sua extensão e criação de diferentes aplicações que requeiram comportamento inteligente. Para isto deve-se adotar uma arquitetura com alta reusabilidade de código; (b) *Foco em jogos digitais:* mesmo com a possibilidade de ser usada em outras aplicações, tal motor de IA está voltado para os diferentes gêneros/estilos de jogos digitais; (c) *Agentes inteligentes:* adoção do paradigma orientado a agentes. Portanto deve-se modelar a percepção, a tomada de decisão e os atuadores de cada um destes agentes; (d) *Homogeneidade:* facilidade para integrar com os demais componentes de um motor de jogo digital, tais como os módulos de renderização e simulação física. Para isto será adotada uma arquitetura dirigida à eventos para efetuar a comunicação entre módulos externos e uma arquitetura dirigida à mensagens para comunicação entre os agentes do motor; (e)

Fácil configuração: a criação e configuração do comportamento dos agentes deve ser bastante simples evitando as especificidades de uma linguagem de programação específica e detalhes referentes a um determinado motor de jogo digital; (f) *Algoritmos de IA:* oferecer um conjunto de algoritmos básicos de IA, tais como definição de trajetórias e algoritmos de busca. Além disso, deve oferecer algoritmos avançados de IA, tais como diferentes algoritmos de Redes Neurais Artificiais, Algoritmos Genéticos, Lógica Difusa, Redes Bayesianas, entre outros algoritmos de aprendizagem de máquinas. Esta diversidade é importante para permitir a configuração de diferentes comportamentos; e (g) *Multi-plataforma:* é fundamental que o motor de IA seja integrável com motores de jogos desenvolvidos para diferentes plataformas. Deve ser compatível com Microsoft Windows e Linux, além do *profile* MIDP 2.0 (J2ME).

Partindo-se destes requisitos estabeleceu-se um conjunto de funcionalidades que deveriam ser implementadas inicialmente pelo motor: (a) permitir projetar facilmente novos sensores para serem usados pelos agentes, tais como sonares, GPS, infravermelho, *line of sight* (LOS), *radius of sight* (ROS), entre outros; (b) flexibilidade para escolher os controladores inteligentes para os agentes, que podem ser simples máquinas de estados (FSM) até uma combinação de sofisticadas técnicas de Aprendizado de Máquinas; (c) implementar algoritmos básicos de Inteligência Artificial, tais como, definição de trajetórias (A*), busca em espaço de estados e min-max; (d) adotar o XML como forma de padronização dos formatos. Permitir que sejam criados conversores que interpretam XML e geram outros formatos mais adequados as especificidades de determinada plataforma computacional; (e) adotar o *Python* como linguagem de scripts para configuração do motor e programação dos agentes inteligentes. Inclusive oferecer a possibilidade de compilar estes *scripts* para serem usados em diferentes linguagens e plataformas. Veja a Figura 1.

Na Figura 1 está especificado um esquema genérico referente a interdependência do motor de IA e seus arquivos de configuração. Toda a configuração e especificação dos comportamentos do motor serão feitos usando XML e *Python*. Basicamente o Python tratará de aspectos comportamentais e o XML descrições de objetos.

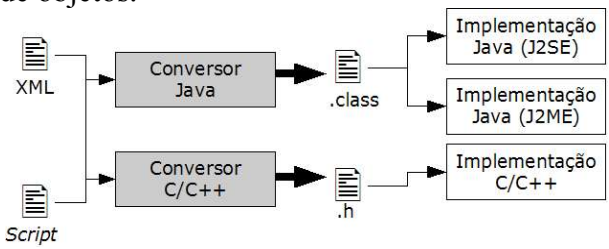

Figura 1. Configuração do motor de IA.

Entretanto estes arquivos poderão ser convertidos para outros formatos para serem utilizados em diferentes plataformas. Se considerar a plataforma móvel, por exemplo, não se pode consumir recursos computacionais escassos para fazer *parsers* XML e interpretar *scripts*. Desta forma torna-se necessário converter esses arquivos para um formato com maior desempenho. Entretanto a forma de especificar comportamentos inteligentes é padronizada e desta forma para o projetista simplifica-se este processo de definição. Para a construção de uma arquitetura computacional capaz de atender os requisitos listados anteriormente e permitir que as funcionalidades iniciais do motor fossem implementadas baseou-se no modelo proposto por Bittencourt [9] para o JFRoGE e na forma de definir agentes utilizada no GNU Mages [5].

Na Figura 2, especifica-se o esquema geral que será adotado na arquitetura do motor de IA. Primeiramente destaca-se que esse motor é um componente externo ao motor de jogo propriamente dito que poderá reagir as ações do motor de IA através do tratamento de eventos. O motor de jogo recebe as entradas dos dispositivos de entrada do sistema (teclado/*mouse*/*joystick*), modifica o estado do

avatar controlado pelo jogador, aplica as regras do jogo sob o domínio e solicita a ação dos agentes controlados pelo computador.

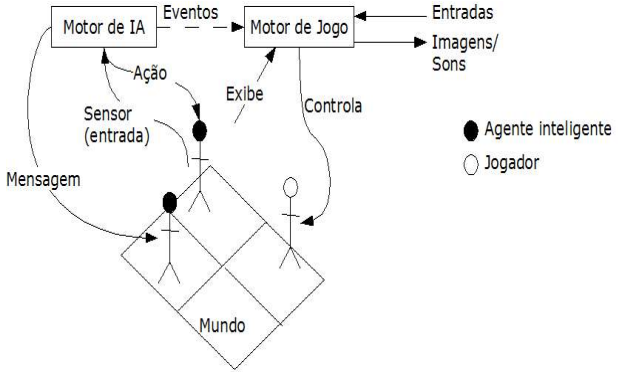

Figura 2. Funcionamento da arquitetura

Cada agente implementa as interfaces computacionais do motor de jogo e do motor de IA, logo torna-se uma interface entre os motores. Cada agente solicita, uma ação para o motor enviando para este um conjunto de informações capturadas pelos seus sensores. O motor de IA então efetua o processamento inteligente usando seus algoritmos de IA e responde para o agente com uma ação. Para comunicação entre agentes adota-se um mecanismo de troca de mensagens.

4. Considerações Finais

Atualmente o presente projeto encontra-se em fase de finalização da revisão da arquitetura e na seqüência será feita a implementação do mesmo adotando as práticas de desenvolvimento de software *eXtreme Programming* (XP) [10, 11]. É importante destacar que este projeto é oriundo de uma parceria entre universidade-empresa, portanto a consolidação desse projeto em um produto representa um incremento da competitividade da empresa e indiretamente uma melhoria para o setor. Além disso, a bolsa de pesquisa concedida para o projeto permitiu que um aprendiz da graduação de Desenvolvimento de Jogos e Entretenimento Digital da UNISINOS tivesse uma experiência de Pesquisa & Desenvolvimento no âmbito de uma empresa desenvolvedora de entretenimento digital.

4. Agradecimentos

É importante agradecer a Federação das Indústrias do Estado do Rio Grande do Sul (FIERGS), em parceria com o Instituto Euvaldo Lodi (IEL/RS) e o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo fomento a pesquisa e desenvolvimento de micro e pequenas empresas em parceria com a universidade.

5. Referências Bibliográficas

[1] DeLoura, M. (ed.). *Game Programming Gems 2*, 3, 4 Charles River Media, Hingham, Massachusetts, 2001.

[2] Fayad, Mohamed E.; Schmidt, Douglas C.; Johnson, Ralph E. *Implementing application frameworks: object-oriented frameworks at work.* New York: John Wiley & sons., 1999.

[3] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design patterns : elements of reusable object-oriented software*. Reading: Addison-Wesley, 1995.

[4] Grand, Mark. *Patterns in java: A catalog of reusable design patterns illustrated with uml*. New York: John Wiley & Sons, 1998.

[5] Bittencourt, João Ricardo; *Ambiente para Simulação de Múltiplos Agentes Autônomos, Cooperativos e Competitivos.* Informática – TCC – Unisinos, São Leopoldo. 2002.

[6] OpenAI. [online]. Disponível em: <http://openai.sourceforge.net/>, Acesso em: 10 set. 2005.

[7] Weka. [online]. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>, Acesso em: 10 set. 2005.

[8] DirectIA. [online]. Disponível em: <http://www.masa-sci.com/directia.htm>, Acesso em: 10 set. 2005.

[9] Bittencourt, João Ricardo; *Um framework para criação de jogos computadorizados multiplataforma.* Dissertação de Mestrado, Informática – PUCRS. 2004.

[10] Beck, Kent. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, Oct. 1999.

[11] eXtreme Programming. [online]. Disponível em: <http://www.extremeprogramming.org/>, Acesso em: 10 set. 2005.