

Especificação do Comportamento de Agentes Virtuais Inteligentes por Demonstração

Gênesis B. Campos

Vladimir O. Di Iorio

Carlos R. Marques Jr.

Alcione P. Oliveira

José Luís Braga

Departamento de Informática – Universidade Federal de Viçosa – Brasil

Resumo

A possibilidade de testar hipóteses e experimentar a realidade sem as despesas e perigos do mundo real fez com que a utilização de ambientes virtuais atraísse a atenção de diversos profissionais. Tão importante quanto interagir em ambientes realistas é lidar com personagens que transmitam a sensação de vida, seres “inteligentes”. A definição da camada de seleção de ações é determinante para o realismo do personagem e pode ser identificada a partir da exemplificação por parte do usuário do comportamento que deve ser executado pelo agente. Diante da não exploração de tal característica pelas abordagens de programação por usuários finais existentes, este trabalho apresenta uma nova proposta de programação por usuários finais para a especificação de agentes virtuais inteligentes. Demonstrando o comportamento através de um avatar imerso no ambiente virtual, o usuário poupa esforços na criação do conjunto de regras que determinam o comportamento do agente. Desta forma, espera-se facilitar o acesso de usuários finais ao desenvolvimento de ambientes virtuais mais realistas e difundir a utilização deste recurso.

Palavras-chave: Agentes Virtuais Inteligentes, Programação por Usuários Finais, Programação por Demonstração

Contatos: {genbarcam, vladimir.iorio}@gmail.com

1 Introdução

A diversidade de possibilidades oferecidas por ambientes virtuais (VEs, do inglês *Virtual Environments*) – simuladores, jogos, entre outros –, sem os riscos e custos inerentes de atividades no mundo real [Angros et al. 2002; Anastassakis et al. 2001], cria uma ótima opção para testar hipóteses e visualizar ideias. Melhor do que simplesmente olhar para símbolos, fórmulas e conceitos, trabalhar com esses ambientes permite ao usuário experimentar a realidade por trás destas definições.

Especialistas em suas respectivas áreas, com necessidades computacionais e que querem fazer sério uso dos computadores, mas que não estão interessados em tornarem-se programadores profissionais, referenciados na literatura como *Usuários Finais* [Nardi 1993], foram atraídos pelos benefícios da utilização de VEs. Surgiram então ambientes virtuais genéricos, sistemas adaptáveis e que, consequentemente, atenderiam a várias demandas.

A *Programação por Usuários Finais* (EUP, do inglês *End-User Programming*) [Myers et al. 2006; Cypher et al. 1993; Nardi 1993; Lieberman 2001], ao contrário da abordagem tradicional, que tenta fazer com que o usuário aprenda a representação aceita pelo computador, preza por tornar o sistema mais próximo do usuário, aceitando representações que não demandem tanta abstração. Buscando romper a barreira entre usuários finais e a especificação de sistemas de simulação, várias técnicas de EUP já foram utilizadas.

Os apelos visuais dos ambientes virtuais, através de gráficos super realistas, tornam-se pouco interessantes na falta de algo dinâmico e comportamental [Aylett and Cavazza 2001]. Agentes Virtuais Inteligentes (IVAs, do inglês *Intelligent Virtual Agents*) são agentes autônomos com representação física, parte de algum ambiente, e têm sido muito utilizados para dar mais realismo a VEs [Reynolds 1999; Brom 2006; Aylett and Cavazza 2001]. IVAs podem ser tutores em softwares educativos, inimigos e/ou aliados em um jogo ou mesmo personagens em animações [Brom 2006].

As técnicas de EUP exploradas para o desenvolvimento de VEs também foram aplicadas para facilitar a implementação de IVAs. O estudo da utilização das abordagens existentes, realizado

neste trabalho, possibilitou a identificação dos pontos fortes e fracos dos mecanismos. Todavia, a especificação do comportamento de IVAs apresenta uma particularidade ainda não explorada pelos sistemas existentes. Depois da construção do ambiente, definição do avatar e das formas de interação dos IVAs entre si e com o VE, e representação das emoções do IVA, basta que o desenvolvedor especifique o comportamento dos personagens. Desta forma, percebe-se que a especificação do comportamento de IVAs pode ser feita utilizando exemplos demonstrados pelo usuário. Em [Reynolds 1999] a especificação do comportamento foi segmentada em (1) seleção de ações, (2) pilotagem e (3) locomoção. A abordagem deste trabalho é a camada de seleção de ações. Entende-se especificação do comportamento como a determinação das ações a serem executadas pelo IVA.

O objetivo deste trabalho é facilitar o acesso de usuários finais ao desenvolvimento de ambientes virtuais através da proposição de uma nova abordagem em EUP para a especificação do comportamento de IVAs. Além de agregar valor na definição de VEs mais interativos, os conceitos aplicados na proposta deste trabalho podem também ser transferidos para o mundo real. A programação de máquinas industriais ou mesmo de robôs domésticos, são exemplos de atividades que podem ser beneficiadas pela nova abordagem.

2 Trabalhos Relacionados

Várias técnicas de EUP já foram empregadas na busca pela facilitação da especificação do comportamento de IVAs por usuários finais. A configuração de parâmetros é a abordagem utilizada na criação de *bots* (abreviação de *robots*) no Counter Strike¹ [Cole et al. 2004]. Esta técnica possibilita ao usuário ativar ou desativar opções que afetam o comportamento da aplicação fazendo uso somente de elementos da interface.

Linguagens de Script são comumente simplificações de linguagens de programação (LPs) tradicionais e possuem sintaxe personalizada para tratar os objetos e ações do domínio. Utilizada em muitos jogos e simuladores como: Robocode²; Gamebots³ e Unreal Script que rodam sobre a plataforma do Unreal Tournament⁴ (UT) [Cole et al. 2004]; e ambiente Logo [Kelleher and Pausch 2005].

As Linguagens Visuais Baseadas em Regras são aquelas nas quais o usuário determina as condições, e para cada uma destas, as ações associadas. O jogo Warcraft III⁵ utiliza esta técnica [El-Nasr and Smith 2006]. A incorporação de mecanismos de programação por demonstração (PBD, do inglês *Programming by Demonstration*), em que o usuário utiliza os elementos visuais da aplicação para a criação dos condicionais é uma variante desta técnica. Agentsheets [Repenning 2000] e KidSim [Cypher and Smith 1995] são ambientes que a implementam.

Bimbo e Vicario em [Bimbo and Vicario 1995] e McDaniel e Myers em [McDaniel and Myers 1998] implementam soluções que utilizam a técnica Programação por Demonstração. Nesta, através do uso de técnicas de inteligência artificial, é criado um programa generalizado a partir de demonstrações realizadas pelo usuário.

O estudo das técnicas de EUP utilizadas para a especificação do comportamento de IVAs revela particularidades que comprometem o desempenho, ou mesmo restringem o acesso dos usuários finais. O número de parâmetros que se pode esperar que o usuário esteja

¹Counter Strike (Valve Corporation) - www.counter-strike.net

²Robocode (Open Source) - robocode.sourceforge.net

³Gamebots (Open Source) - sourceforge.net/projects/gamebots

⁴Unreal Tournament (Epic Games) - www.unrealtournament3.com

⁵Warcraft III (Blizzard) - <http://www.blizzard.com/us/war3/>

apto a entender ao adotar a configuração de parâmetrose é bastante pequeno e consequentemente compromete a implementação de comportamentos que atendam às suas necessidades [Barbosa 1999]. O emprego de linguagens de script, ainda que muito mais simples que as LPs convencionais, assim como a utilização dos ícones propostos pelas linguagens de fluxo de controle, que não dispensam o entendimento da semântica da estrutura, necessitam de um mínimo conhecimento de programação que muitas vezes falta ao usuário final. Por sua vez, as LPs visuais baseadas em regras exigem a criação de condicionais basicamente para cada interação do agente com o meio, tornando a técnica inadequada para a especificação de comportamentos mais elaborados ou quando o agente precisa interagir com um ambiente complexo. Finalmente, os sistemas que incorporam a PBD restringem a atuação do usuário e demandam nova demonstração caso ocorram inferências errôneas nas exemplificações anteriores. Desta forma, vê-se que as técnicas de EUP utilizadas para a especificação do comportamento de IVAs são passíveis de questionamento quanto à real facilitação do trabalho do usuário. Portanto, percebe-se que mais pode ser feito no intuito de minimizar os problemas encontrados.

3 Programação do Comportamento de Agentes por Demonstração

A PBD é, por motivos óbvios, a mais adequada ao problema. Especificar comportamentos por demonstração é o mesmo que ensinar pelo exemplo. Todavia, o encapsulamento da solução gerada a partir da demonstração vai em desencontro com a usabilidade das abordagens que seguem esse mecanismo. Uma vez que o usuário não programa explicitamente, o comportamento inferido pelo sistema a partir da demonstração pode não estar de acordo com aquilo que era esperado, sendo necessário, portanto, realizar modificações na solução proposta. Nos sistemas encontrados na literatura, essas modificações são possíveis somente via nova demonstração. Tal solução, além de não economizar o tempo do usuário, pode causar insatisfação, uma vez que não assegura a correção do problema, expondo-o à necessidade de novo retrabalho.

As técnicas vistas isoladamente possuem pontos que as qualificam como alternativas para a solução do problema abordado neste trabalho. Contudo, existem também características que comprometem sua utilização. Observa-se, também, que a aplicação de mecanismos diferentes para resolução do problema não é conflitante, e sim complementar. Recursos seguindo a PBD resolvem a manipulação de variáveis textuais nas LPs baseada em regras, que por sua vez, permitem a modificação do comportamento, sem necessidade de nova demonstração.

Vista a complementaridade dos diferentes mecanismos, propõe-se a nova abordagem de EUP para a especificação do comportamento de IVAs. A programação visual de regras, com toda sua expressividade e seu fácil entendimento e manipulação, é a alternativa utilizada neste trabalho para contornar o encapsulamento das soluções geradas pela PBD.

Essa aliança caracteriza o caminho reverso da variante da programação visual de regras que utiliza os elementos visuais da aplicação para a criação dos condicionais. Ao invés de permitir que o usuário crie as regras exemplificando as situações que serão vividas no ambiente, nesta nova solução o usuário demonstra, através de um avatar imerso no VE, como o IVA deve se comportar. As ações tomadas pelo usuário diante das situações casualmente enfrentadas, resultam em regras de comportamento que serão seguidas pelo IVA, e nesta proposta podem ser posteriormente editadas sem nova demonstração. A abordagem permite que, por exemplo, um *bot* de um jogo de tiro de primeira pessoa seja gerado a partir do simples registro do jogo de um usuário.

A incorporação das LPs baseadas em regras à PBD traz consigo os problemas existentes neste mecanismo. Uma vez que as regras serão geradas automaticamente, não é necessário preocupar-se com o quão maçante seria criá-las, como discutido em [Coura et al. 2006], nem tão pouco com a possibilidade de geração de regras conflitantes. Contudo, sistemas que utilizam este mecanismo normalmente se atêm à especificação de comportamentos simples, sem muitas variáveis a serem consideradas, posto que uma grande

quantidade de regras dificultaria aos usuários entender o que foi implementado. A proposta deste trabalho busca atender a todo o tipo de comportamento, simples ou complexo, e sobretudo garantir que usuários finais possam utilizá-la. Portanto, uma nova adaptação se fez necessária para assegurar a compreensão do usuário.

O conjunto de variáveis a ser considerado para a especificação de comportamentos complexos é muito extenso. Seguindo a ideia da configuração de parâmetros, este trabalho incorpora ao mecanismo de PBD os atributos que serão levados em conta para a criação das regras. Para cada atributo, também deve ser definido um conjunto padrão de valores, que pode ser posteriormente editado conforme o interesse do usuário. A coleção de atributos definidos estará disponível ao usuário, e a cada um estará associado seu respectivo elemento visual (se existir) e descrição. Antes de iniciar a demonstração o usuário deve definir quais atributos ele deseja que sejam considerados para a criação do comportamento. A ideia deste recurso é permitir que o usuário dose, conforme seu grau de entendimento, a quantidade de variáveis a serem manipuladas, permitindo maior predição do comportamento que será executado pelo IVA a partir da análise das regras geradas.

A quantidade de regras é outro fator crítico da solução proposta. Por serem geradas a partir de arranjos de atributos, o número de expressões condicionais resultante da geração automática é imenso. Visto o problema, além das regras foram também adotadas árvores de decisão como forma de representação do conhecimento. Os nós de cada nível representam um atributo e os ramos destes fazem o papel dos valores associados. Acredita-se que a utilização desta técnica pode contribuir muito para o melhor entendimento do usuário, uma vez que ao invés de ser exposta uma lista de regras, ainda que ordenadas, o usuário pode navegar entre os nós da árvore até alcançar aquela que represente a situação que ele deseja analisar. A técnica ainda agrega valor à solução ao impedir que sejam geradas expressões com diferença na quantidade de atributos testados, dispensando a necessidade de adoção de políticas de prioridade de execução para comportamentos vinculados a expressões mutuamente verdadeiras. Outro benefício está na avaliação das expressões condicionais. Enquanto regras não organizadas utilizando esta técnica precisam ser testadas uma a uma, algoritmos podem otimizar o posicionamento dos atributos nos níveis da árvore, de forma a definir qual o comportamento deve ser executado testando a menor quantidade possível de expressões.

3.1 Formalização da abordagem

A seleção de ações é o meio através do qual um IVA resolve o problema de decidir o que fazer, qual atitude tomar. É a parte executiva da inteligência dos agentes [Bryson 2004]. Sua análise permite prever o comportamento de determinado agente sem ser necessário vê-lo agir. Segundo Bryson a adoção de padrões – planos reativos básicos⁶, planos reativos POSH⁷, máquinas de estado finitas e determinismo ambiental⁸ – para a modelagem da camada de seleção de ações simplifica a solução e consequentemente facilita o entendimento do usuário.

Planos reativos básicos e POSH demandam noções de conceitos como prioridade e paralelismo, que podem não ser conhecidos pelo usuário e colocam em risco a compreensão da camada de seleção de ações. Máquinas de estado finitas oferecem uma abordagem mais palpável ao usuário. Todavia, por focar nas ações, possuem um conjunto de regras diferentes para cada estado, o que poderia dificultar a predição do comportamento que será executado. Determinismo ambiental, o padrão adotado neste trabalho, orienta-se nos eventos, discretiza o conjunto de situações que o agente pode presenciar e associa a cada uma delas a ação que deve ser tomada. A simplicidade do conjunto de regras, produto do determinismo ambiental, motivaram a escolha deste trabalho.

Tão importante quanto a escolha da PBD como estratégia de aprendizado, é a forma com que o sistema representa o conhecimento adquirido. A modelagem dos eventos como regras é intrínseca

⁶Basic Reactive Plans

⁷Parallel-rooted, Ordered, Slip-stack Hierarchical

⁸Environmental Determinism

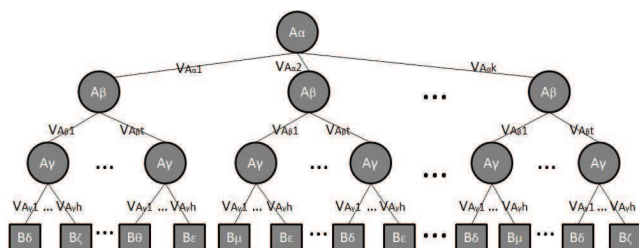


Figura 1: Exemplo da camada de seleção de ações em esquema híbrido: regras e árvore de decisão.

à adoção de LPs baseadas em regras, todavia, a quantidade de expressões condicionais motivaram a opção por um esquema híbrido: regras e árvores de decisão como representação do conhecimento presente na camada de seleção de ações.

Portanto, o comportamento gerado é formado pela tupla (A, B) . Onde: A representa conjunto de atributos selecionados pelo usuário para a criação das regras, $A = \{A_1, A_2, \dots, A_n\}$; e B , por sua vez, simboliza o conjunto de comportamentos demonstrados, $B = \{B_1, B_2, \dots, B_k\}$. A é subconjunto de R , biblioteca de atributos que podem ser considerados pelo usuário e definidos previamente pelo desenvolvedor da aplicação. Cada atributo pertencente à biblioteca R possui seu conjunto de valores associado $V_{R_i} = \{V_{R_i1}, V_{R_i2}, \dots, V_{R_im_{R_i}}\}$, que podem ser definidos pelo desenvolvedor ou editados conforme a necessidade do usuário. Não necessariamente, todos os comportamentos do conjunto B estarão inicialmente associados às conjunções entre os atributos, alguns deles podem estar disponíveis apenas para futura edição.

Um exemplo genérico do comportamento gerado pode ser visto na Figura 1. Neste caso, tem-se o conjunto de atributos selecionados pelo usuário sendo $A = \{A_\alpha, A_\beta, A_\gamma\}$ com seus respectivos conjuntos de valores: $V_{A_\alpha} = \{V_{A_\alpha1}, V_{A_\alpha2}, \dots, V_{A_\alpha k}\}$, $V_{A_\beta} = \{V_{A_\beta1}, \dots, V_{A_\beta t}\}$ e $V_{A_\gamma} = \{V_{A_\gamma1}, \dots, V_{A_\gamma h}\}$. Os comportamentos exibidos na árvore: $B_\delta, B_\zeta, B_\theta, B_\epsilon$ e B_μ pertencem ao conjunto B , mas não se pode dizer que B possui somente estes, outros podem estar disponíveis para posterior edição. A leitura de uma das expressões da árvore exemplificada é: Caso A_α seja igual a $V_{A_\alpha1}$ e A_β igual a $V_{A_\beta1}$ e A_γ igual a $V_{A_\gamma1}$, então execute o comportamento B_δ .

4 Ambiente de testes

Embora existam diversos sistemas de criação de simulações genéricas, Agentsheets [Repenning 2000] e KidSim [Cypher and Smith 1995] por exemplo, a utilização destes para o desenvolvimento da plataforma de testes é impossível, pois seriam necessárias modificações e estes são proprietários. O custo de desenvolvimento de um VE que incorporasse os recursos visuais dos ambientes atuais é enorme. Entretanto, a modularização do desenvolvimento de jogos de computador [El-Nasr and Smith 2006] permite que estes sejam adaptados para atender às necessidades do trabalho.

Contudo, as adaptações que podem ser realizadas dependem das possibilidades oferecidas pelo jogo. El-Nasr e Smith, em [El-Nasr and Smith 2006], estudaram a utilização de várias linguagens de extensão e a *Unreal Script*, linguagem oferecida pelo UT, destacou-se por permitir a customização de praticamente todo o jogo. A expressividade da *Unreal Script* foi determinante para a escolha da plataforma de implementação da ferramenta de validação.

A extensa capacidade de personalização permitiu que o UT fosse utilizado para pesquisas em diversas áreas, e para diversos públicos. Na área de especificação de agentes virtuais inteligentes pode-se citar: Gamebots [Kaminka et al. 2002], Indigente [Monteiro 2006] e Pogamut 2 [Burkert et al. 2007]. O primeiro permite o controle de personagens do UT em arquitetura cliente-servidor, enviando informações sensoriais para o cliente (que pode ser um IVA ou um jogador), e repassando ao servidor do jogo quais os comandos foram executados pelos agentes. Embora ofereça recursos para o desenvolvimento de agentes, a comunicação por mensagens, a quantidade de trabalho de baixo nível que precisa ser realizado an-

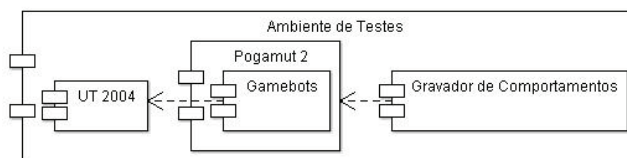


Figura 2: Diagrama de componentes do ambiente de testes

tes de realmente iniciar a especificação dos IVAs, e a não existência de uma arquitetura de agentes pré-definida são barreiras mesmo para programadores mais experientes. Os demais, oferecem exatamente a solução para os problemas enfrentados no Gamebots. Ambos encapsulam-no, oferecendo uma camada de alto nível para a especificação dos IVAs, e uma arquitetura para a modelagem de agentes. A grande quantidade de informação disponível online, graças à vasta rede de usuários, o suporte por parte dos desenvolvedores e a continuidade do desenvolvimento, justificam a escolha do Pogamut 2 para a implementação da plataforma de testes.

Portanto, a plataforma para a implementação do ambiente de testes da nova abordagem de EUP proposta será o jogo UT 2004, devido a sua flexibilidade de adaptação e o arcabouço Pogamut 2, justificado acima. O diagrama de componentes do ambiente de testes é apresentado na Figura 2.

4.1 Vista Geral do Funcionamento do Sistema

A ideia básica da aplicação a ser implementada é permitir que o usuário jogue UT 2004 e que a partir do comportamento por ele executado, um IVA seja gerado. Na interface do ambiente de testes, o usuário se conectará ao servidor de um jogo previamente criado no UT. Neste, será inserido um agente marionete, que será comandado pelo usuário diante das situações vividas no ambiente. Deverá ser solicitada, então, a geração das regras a partir do registro da demonstração. A camada de seleção de ações será disponibilizada e o usuário terá a opção de editá-la. Satisfeito com as expressões resultantes, restará conectar o IVA padrão no ambiente, que interpretará as regras geradas e executará um comportamento semelhante ao que foi demonstrado.

5 Comparativo com outras técnicas

Buscando apresentar os benefícios da proposta deste trabalho em relação a outras técnicas de programação para usuários finais já utilizadas para o mesmo fim, discute-se hipoteticamente a criação do comportamento de IVAs através da configuração de parâmetros, LPs visuais baseadas em regras e programação por demonstração. Embora as linguagens de script tenham, também, sido apresentadas como opções já disponibilizadas aos usuários finais para a realização da tarefa, estas estão muito aquém da facilitação das demais e, por isso, foram preteridas nesta análise.

Em sistemas que utilizam a configuração de parâmetros quanto maior o número de regras mais realista tende a ser o agente e mais tempo será demandado para a implementação. Fica claro que esta alternativa não passa da disponibilização da camada de seleção de ações ao usuário final, também oferecida neste trabalho. Todavia, as customizações se resumem aos valores comparados nas expressões condicionais. Não é possível alterar a quantidade de regras, o que pode dificultar o entendimento, como já discutido anteriormente. Outro problema é a impossibilidade de criar novas regras com ações distintas, ou mesmo editar as regras pré-estabelecidas. Portanto, a configuração de parâmetros é nada além da possibilidade de alterar o conjunto padrão de valores de cada um dos atributos disponibilizado na biblioteca do estudo de caso, recurso que, embora não implementado, já faz parte da proposta deste trabalho. Logo, percebe-se que, a não necessidade de interferência manual do usuário em todas as regras, evitada pela demonstração, e a possibilidade de alteração das ações associadas às regras são melhorias relevantes da nova abordagem em relação a configuração de parâmetros.

As LPs visuais baseadas em regras, principalmente as que utilizam os elementos visuais da aplicação para a criação dos condi-

cionais, são as mais utilizadas na literatura para a especificação do comportamento de agentes. Por basearem-se na interface do VE no qual interagem, são mais intuitivas e consequentemente mais acessíveis aos usuários finais. Tradicionalmente, as LPs visuais baseadas em regras trabalham com a interação do agente com o meio sem a necessidade de analisar propriedades internas do mesmo. No caso da criação de um IVA, tal qual o que será desenvolvido neste estudo de caso, a análise de atributos é indispensável. Então, a criação das regras envolveria a análise dos que serão disponibilizados na biblioteca, cada qual com três faixas de valores em média. A quantidade de regras a ser gerada é imensa, o que torna a técnica inadequada, uma vez que seria maçante ao usuário ter de criar todo o conjunto de expressões manualmente. Não obstante, a regras estariam organizadas sequencialmente, dificultando o entendimento e também a eficiência da execução. Portanto, graças a possibilidade de demonstrar o comportamento, a técnica desenvolvida neste trabalho é muito mais eficiente, e devido a organização da camada de seleção de ações em árvore, mais compreensível.

Por sua vez, a criação de comportamentos através da programação por demonstração é a que mais se aproxima à técnica desenvolvida neste trabalho. Em relação às propostas implementadas em [Bimbo and Vicario 1995] e [McDaniel and Myers 1998], o não ocultamento da camada de seleção de ações, permitindo criação, ou edição, de regras sem necessidade de nova demonstração, e a possibilidade de demonstrar o comportamento jogando, e não em situações específicas ou ciclos, incorporando o fator diversão à tarefa, agregam valor à abordagem deste trabalho.

6 Conclusões e Trabalhos Futuros

Embora a especificação do comportamento do IVAs através da demonstração já tenha sido proposta na literatura, o estudo realizado neste trabalho das diversas abordagens de EUP, voltadas, ou não, à tarefa em questão, possibilitou adaptações à técnica existente. Estas fazem com que a nova proposta, comparada, hipoteticamente, aos mecanismos já utilizados para tal propósito, facilite ainda mais o acesso a sistemas de simulação a profissionais de diversas áreas, antes privadas dos benefícios desta tecnologia.

Ainda que o trabalho consista na proposta de uma nova abordagem de programação para usuários finais, a implementação do ambiente de testes é fundamental para a realização de testes e consequentes análises comparativas quantitativas em relação à satisfação e desempenho do usuário na utilização desta e das outras técnicas já aplicadas à mesma tarefa. Pretende-se desenvolver a aplicação como parte da interface de desenvolvimento integrado do Pogamut 2. Desta forma sua utilização seria difundida e desenvolvimento compartilhado junto a aqueles que faz uso do arcabouço, possibilitando maior contribuição do trabalho à comunidade científica.

Agradecimentos

À CAPES e FAPEMIG, pelo apoio financeiro.

Referências

- ANASTASSAKIS, G., RITCHINGS, T., AND PANAYIOTOPOULOS, T. 2001. Multi-agent systems as intelligent virtual environments. In *KI '01: Proceedings of the Joint German/Austrian Conference on AI*, Springer-Verlag, UK, 381–395.
- ANGROS, JR., R., JOHNSON, W. L., RICKEL, J., AND SCHOLER, A. 2002. Learning domain knowledge for teaching procedural skills. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM, New York, NY, USA, 1372–1378.
- AYLETT, R., AND CAVAZZA, M. 2001. Intelligent virtual environments - a state-of-the-art report. In *Eurographics'01*.
- BARBOSA, S. D. J. 1999. *Programação via Interface*. PhD thesis, Departamento de Informática-Pontifícia Universidade Católica do Rio de Janeiro.
- BIMBO, A. D., AND VICARIO, E. 1995. Specification by-example of virtual agents behavior. *IEEE Transactions on Visualization and Computer Graphics* 1, 4, 350–360.
- BROM, C., 2006. Action selection for virtual humans in large environments. Doctoral Thesis Abstract – Faculty of Mathematics and Physics – Department of Software and Computer Science Education – Charles University in Prague.
- BRYSON, J. J. 2004. Action selection and individuation in agent based modelling. In *Proceedings of Agent 2003: Challenges in Social Simulation*, Argonne National Laboratory, 317–330.
- BURKERT, O., KADLEC, R., GEMROT, J., BÍDA, M., HAVLÍČEK, J., DÖRFLER, M., AND BROM, C. 2007. Towards fast prototyping of ivas behavior: Pogamut 2. In *IVA '07: Proceedings of the 7th international conference on Intelligent Virtual Agents*, Springer-Verlag, Berlin, Heidelberg, 362–363.
- COLE, N., LOUIS, S. J., AND MILES, C. 2004. Using a genetic algorithm to tune first-person shooter bots. In *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, New York, NY, USA, vol. 1, 139–145.
- COURA, D. P., IORIO, V. O. D., LIMA, A. G., OLIVEIRA, A. P., AND ANDRADE, M. V. A. 2006. Animações através de programação por demonstração. In *IHC '06: Proceedings of VII Brazilian symposium on Human factors in computing systems*, ACM, New York, NY, USA, 81–90.
- CYPHER, A., AND SMITH, D. C. 1995. Kidsim: end user programming of simulations. In *CHI '95: Proceedings of the SIG-CHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co., NY, USA, 27–34.
- CYPHER, A., HALBERT, D. C., KURLANDER, D., LIEBERMAN, H., MAULSBY, D., MYERS, B. A., AND TURRANSKY, A., Eds. 1993. *Watch what I do: programming by demonstration*. MIT Press, Cambridge, MA, USA.
- EL-NASR, M. S., AND SMITH, B. K. 2006. Learning through game modding. *Comput. Entertain.* 4, 1.
- KAMINKA, G. A., VELOSO, M. M., SCHAFFER, S., SOLLITTO, C., ADOBBATI, R., MARSHALL, A. N., SCHOLER, A., AND TEJADA, S. 2002. Gamebots: a flexible test bed for multiagent team research. *Commun. ACM* 45, 1, 43–45.
- KELLEHER, C., AND PAUSCH, R. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.* 37, 2.
- LIEBERMAN, H., Ed. 2001. *Your wish is my command: programming by example*. Morgan Kaufmann Publishers Inc., CA, USA.
- MCDANIEL, R. G., AND MYERS, B. A. 1998. Building applications using only demonstration. In *IUI '98: Proceedings of the 3rd international conference on Intelligent user interfaces*, ACM, New York, NY, USA, 109–116.
- MONTEIRO, I. M., 2006. IAF - indigente agent framework: Um framework para o desenvolvimento de agentes cognitivos em jogos de primeira pessoa. Monografia de Graduação – Depto. de Ciência da Computação – Instituto de Matemática – UFBA.
- MYERS, B. A., KO, A. J., AND BURNETT, M. M. 2006. Invited research overview: end-user programming. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, ACM, New York, NY, USA, 75–80.
- NARDI, B. A. 1993. *A small matter of programming: perspectives on end user computing*. MIT Press, Cambridge, MA, USA.
- REPENNING, A. 2000. Agentsheets: an interactive simulation environment with end-user programmable agents. In *Proceedings of the IFIP Conference on Human Computer Interaction*, 1–8.
- REYNOLDS, C. 1999. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference 1999*, Miller Freeman Game Group, 763–782.