Metodologia de Desenvolvimento de Jogos Sérios: especificação de ferramentas de apoio *open source*

Rafaela Vilela da Rocha¹, Aparecida Maria Zem-Lopes^{1,2}, Laís Zagatti Pedro¹,
Ig Ibert Bittencourt³, Seiji Isotani¹

¹Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
(USP) – São Carlos – SP – Brasil

²Faculdade de Tecnologia de Jahu (FATEC Jahu) – Jaú– SP – Brasil

³Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Maceió – AL – Brasil

Abstract. *Serious games are being used ever more for purposes of learning, training and performance evaluation. However, to be successful as a final product, its development has to be systematic and multi-disciplinary. Thus, the choice of support tools impact beyond the financial cost. This paper describes an overview of the iterative and integrative methodology for developing serious games, expanding the specification with open source tools. Appropriate tools are reviewed and selected in order to provide a set of applications that could be used according to serious games design needs.*

Resumo. *O uso de jogos sérios vem crescendo para fins de aprendizado, treinamento e avaliação do desempenho dos seus usuários. Entretanto, para obter sucesso como produto final, seu desenvolvimento tem que ser sistemático e multidisciplinar. Sendo assim, a escolha das ferramentas de suporte impacta muito além do custo financeiro. Este artigo apresenta uma visão geral da metodologia iterativa e integradora para desenvolvimento de jogos sérios com foco na descrição de ferramentas open source. Foram revisadas e selecionadas as ferramentas adequadas, de modo a fornecer um conjunto de ferramentas que poderão ser usadas de acordo com as necessidades do projeto do jogo sério.*

1. Introdução

O desenvolvimento de Jogos Sérios (JSs) é um processo complexo, com alto custo (de recursos humanos, financeiros, tempo e materiais), que requer profissionais qualificados, que detêm conhecimentos de aprendizagem, avaliação, simulação e de jogos em si, além do conteúdo no domínio de aplicação e das ferramentas que serão utilizadas no processo de produção [Aslan e Balci, 2015; Balci, 2012].

Para oferecer apoio ao desenvolvimento de JSs, diferentes metodologias foram propostas [Adams et al. 2007; Akilli e Cagiltay, 2006; Aslan e Balci, 2015; Becker e Parker, 2012; Freitas e Jarvis, 2006; Hays, 2005], além das específicas existentes nas áreas de simulação [Balci, 2012; Banks et al., 2001; IEEE, 2003 e 2010; Ford, 2004], jogo [Chandler, 2012; Novak, 2010; Schuyttema, 2008], e aprendizagem e treinamento [ABNT, 2001; Dick, Carey e Carey, 2004; Kemp, 1977]. Entretanto, de forma geral,

estas metodologias não descrevem todo o ciclo de vida, compreendendo os múltiplos requisitos dos JSs. As específicas enfatizam apenas os elementos de jogo, ou da simulação, ou da instrução. Já as outras, para fins de produção de jogos sérios, jogos e simulações educacionais, que serão apresentadas na seção 2, unificam alguns elementos, mas sem abranger todos os requisitos necessários, criando uma lacuna entre elas. Além disto, nem sempre são especificados os artefatos que devem ser produzidos em cada processo e quais as ferramentas que podem apoiar o trabalho dos envolvidos.

Este artigo apresenta a metodologia iterativa e integradora de desenvolvimento de jogos sérios, com foco nas ferramentas de código livre que podem ser usadas para criar os diferentes artefatos. Na seção 2 é apresentado o estado da arte de metodologias de desenvolvimento, seguida da metodologia em questão (na seção 3), e de ferramentas que são agrupadas por tipo de artefatos gerados (na seção 4). Na seção 5 são apresentados e discutidos os resultados, seguidas das considerações finais (na seção 6).

2. Trabalhos Relacionados

Uma revisão da literatura foi realizada para obter uma compreensão geral do estado da arte sobre as metodologias de desenvolvimento de jogos sérios e afins; e identificar os requisitos e desafios que devem ser abordados e superados durante sua produção. Estas metodologias podem ser classificadas em três grupos: (1) **Específicas** para jogo, ou simulação, ou aprendizagem, conforme referências citadas na Introdução. Elas não abrangem todos os requisitos de JSs, tais como, não linearidade, fidelidade lógica e física da simulação, definição dos objetivos de aprendizagem/ treinamento e conteúdo, avaliação do desempenho do aprendiz, desafios, recompensa, níveis e *feedback* contínuo [Trybus, 2014; Hays, 2005]. (2) Para **simulação e jogo educacional**: Modelo FIDGE [Akilli e Cagiltay, 2006], *Design* da Simulação Educacional [Adams et al, 2007], GAMED [Aslan e Balci, 2015], Processo Sistemático [Hays, 2005] e Modelo DGBL-ID [Zin, Jaafar e Yue, 2009]. Elas não abordam os requisitos de fidelidade lógica e física da simulação, não oferecem apoio para avaliação e *feedback* do desempenho do aprendiz, e não integram jogo-instrução de forma efetiva. E (3) para **jogo sério**: Processo Synergy [Becker e Parker, 2012], Processo baseado no *Design* Centrado no Usuário [Freitas e Jarvis, 2006], Modelo SG-ISD [Kirkley, Tomblin e Kirkley, 2005], Processo baseado em Processo Unificado Rational [Rodrigues, Machado e Valença, 2010] e Modelo de Referência de Greenblat [Van Der Zee, Holkenborgb e Robinson, 2012]. Elas têm foco no *design* instrucional e do jogo, porém sem integração detalhada, não apoiam a avaliação do JS, e apenas descrevem o que deve ser avaliado sobre o desempenho do aprendiz, sem oferecer suporte. Além disto, algumas abordam apenas algumas fases do desenvolvimento [Rodrigues, Machado e Valença, 2010; Freitas e Jarvis, 2006].

De forma geral, essas metodologias nem sempre detalham os artefatos e os profissionais que estão envolvidos em cada processo. Da mesma forma, as ferramentas de suporte se limitam a aplicativos de autoria, sem oferecer opções para os usuários.

Neste contexto, é necessária uma nova metodologia que integre, de fato, as áreas de jogo, simulação, aprendizagem, avaliação e domínio de aplicação. Somado a isso, não há metodologia na literatura que especifica o uso de ferramentas *open source* em cada fase de desenvolvimento de jogos sérios. Estas ferramentas podem viabilizar a produção dos artefatos, inclusive gerá-los automaticamente de forma padronizada, o que pode contribuir para aumentar a qualidade do produto final.

3. Metodologia de Desenvolvimento de Jogos Sérios

Para superar os desafios apresentados, uma nova metodologia iterativa e integradora foi criada por [Rocha, 2014], para apoiar o desenvolvimento de jogos sérios em específico, com a finalidade de treinar e avaliar o desempenho humano. Ela foi avaliada por especialistas em Engenharia de *Software* e desenvolvimento de JSs, e foi utilizada para criar 20 protótipos de simulações interativas e um jogo sério completo. Este JS, para treinamento de combate a vazamento de gás de cozinha, foi utilizado e avaliado pelo Corpo de Bombeiros do Estado de São Paulo. A seguir são descritos os objetivos de cada processo da metodologia, os quais são apresentados na Figura 1.

Figura 1. Metodologia de desenvolvimento de jogos sérios [adotado de Rocha, 2014]

O processo (1) Planejamento (Pré-produção) visa a elaboração do plano inicial, a partir das necessidades de treinamento, contendo a situação do mundo real que será simulada, os procedimentos e as competências que serão treinadas. Este plano deve ser feito pelos responsáveis técnicos, que pode ser um profissional ou uma equipe que contém um ou mais pedagogo, professor, treinador e/ou especialista no domínio.

A etapa de Produção compreende os processos de 2 a 5. Na (2) Análise são especificados os requisitos para criar o jogo sério: incluindo dados do jogo, da simulação, e da arquitetura de suporte (definidos pelo analista de sistema), e do treinamento e avaliação (definidos pelo responsável técnico). Em (3) Projeto, os requisitos de jogo, simulação, arquitetura, treinamento e avaliação são transformados em projetos de jogo (responsabilidade do *designer*), simulação e arquitetura (responsabilidade do analista de sistema), e programa de treinamento (responsável técnico). Em (4) Implementação, os projetos do processo anterior são implementados pela equipe desenvolvedora, apoiados pelo responsável técnico. Esta equipe é formada por um ou mais programador e *designer* de arte e som, ou os respectivos responsáveis pelo desenvolvimento do código-fonte, da arte e do som (que podem incluir desde

modeladores, animadores, produtores de som, até atores de voz e testadores). Em (5) Integração e Teste, os artefatos são integrados e testados até a conclusão do JS. Os profissionais da equipe desenvolvedora realizam suas funções para integrar e testar os recursos desenvolvidos no processo anterior.

Durante a Pós-produção, no processo (6) Execução, os aprendizes realizam o treinamento e, ao final, fazem uma autoavaliação (usando o questionário descrito em [Rocha 2014]). O instrutor é responsável por acompanhar e auxiliar os aprendizes. Já no processo (7) Avaliação, o instrutor é responsável por gerar os relatórios de desempenho, realizar sua avaliação e fornecer o *feedback* aos aprendizes.

Ao final de cada processo, a atividade de verificação e validação deve ser conduzida pela equipe desenvolvedora, em conjunto com o responsável técnico. Caso seja necessário, o desenvolvimento pode voltar em algum processo anterior, em um processo iterativo e incremental.

Além da metodologia, também foram criados artefatos, modelos, componentes e uma arquitetura de suporte que guiam e apoiam cada processo do ciclo de vida. Para criar, modelar ou gerar estes artefatos, diferentes ferramentas devem ser utilizadas, desde as comuns no desenvolvimento de *software* em geral, tais como, editores de texto e sistema de controle de versão, até as mais específicas para a criação do jogo, tais como ferramentas de modelagem 3D e motores de jogos. Dentre as opções que podem ser utilizadas, as principais ferramentas de código livre são apresentadas na próxima seção.

4. Artefatos e Ferramentas de Código Livre

A escolha das ferramentas abrange três plataformas computacionais distintas, em relação ao jogo sério: (1) a que os aprendizes e instrutores utilizarão para executá-lo (cliente); (2) a que ele ficará disponível e armazenará os resultados do treinamento (servidor); e (3) a que os profissionais o produzirão; conforme apresentado na Figura 2.

Figura 2. Visão geral das distintas plataformas computacionais e os conjuntos de ferramentas relacionadas

As duas primeiras são especificadas na atividade de “Análise” (Figura 1, fase 2) de requisitos da “arquitetura de suporte”, referente aos requisitos mínimos e necessários para o cliente (*hardware* e *software*) e o servidor (hospedagem, banco de dados e arquitetura de suporte); e projetadas no processo de “Projeto” (Figura 1, fase 3), referente a atividade de “*design* da arquitetura e banco de dados”. Entretanto, é também necessário o planejamento de quais ferramentas serão utilizadas pelos profissionais para fazer, além do “Planejamento” e da “Análise”, os processos do próprio “Projeto” do JS,

e então a “Implementação” e depois a “Integração e Teste” (Figura 1, fases 4 e 5). Para isto, são consideradas as necessidades dos usuários finais (requisitos da arquitetura). Porém, também é relevante que os profissionais conheçam as opções de ferramentas que podem ser usadas.

Neste contexto, este trabalho contribui estendendo a especificação que foi criada em Rocha [2014], de modo a contemplar as **ferramentas *open source*** que podem apoiar os profissionais durante o uso da metodologia. A escolha das ferramentas foi feita a partir das necessidades dos artefatos que serão criados em cada processo. Foi identificado um conjunto de ferramentas, para cada atividade específica, que possui o código livre e pode ser utilizada em mais de uma plataforma (geralmente são para Windows, Linux e MacOS). Muitas vezes este conjunto abrange o uso de cada ferramenta para um contexto específico, que dependerá das necessidades do projeto.

Nas próximas seções são descritas as ferramentas de *storyboarding*, diagramação UML, modelagem formal e 3D, gerenciamento de banco de dados e motor de jogos, que são essenciais para a produção do JS. As ferramentas de apoio às atividades mais gerais, referentes ao quadro em amarelo na Figura 2, são reportadas em [Rocha 2015]: criação de imagens, edição de áudio e vídeo, criação de ontologias, controle de versão, edição de texto e teste.

4.1. Ferramentas de Apoio ao Projeto do Jogo Sérió

Um dos diferenciais da metodologia criada é a especificação e o uso de modelos integradores como base para o projeto dos JSs, de modo a possibilitar que equipes multidisciplinares projetem a simulação, o jogo e cada uma de suas fases, com as avaliações e *feedbacks* necessários. Isto inclui *storyboards*, que integram a descrição do cenário com as interações; diagrama estruturado, que integra objeto do jogo e simulação; e modelos formais, que integram simulações e avaliações. A seguir são descritas ferramentas que podem ser usadas para criar estes artefatos.

Storyboards são usados para entender e projetar o cenário de treinamento, seus objetos e personagens, e descrever as interfaces de interação e o fluxo de ações que pode acontecer durante o treinamento [Rankin et al., 2011]. Eles apoiam o trabalho colaborativo por serem de fácil entendimento, tanto para os desenvolvedores quanto para o responsável técnico e as outras pessoas envolvidas. As ferramentas de código livre que podem ser usadas são os editores de imagens: **GIMP** [2014]: para imagens *bitmap* (mapa de bits), desenvolvido em C e GTK+; e **Inkscape** [2015]: para imagens vetoriais, desenvolvido em C e C++. Ambos têm versões para Windows, Linux e Mac OS. A diferença é o tipo de imagem a ser criada, sendo que cada ferramenta tem funcionalidades específicas para isto.

Diagramas baseados na Linguagem de Modelagem Unificada (*Unified Modeling Language*TM - UML) devem ser utilizados para representar modelos estruturados e/ou comportamentais. Além disto, um diagrama baseado no diagrama estruturado de classe UML foi especificado em [Rocha, 2014] para integrar diferentes características de um objeto simulado: arte, áudio, programação e interface. Para modelar estes diagramas, podem ser usadas as ferramentas: **Dia** [2013] desenvolvida em C, e **UMLet** [2015] e **ArgoUML** [2009], ambas desenvolvidas em Java.

Modelos formais devem ser utilizados para modelar o comportamento do procedimento de treinamento e os pontos de avaliação do desempenho humano. Em

[Rocha, 2014], foram especificados os processos para a modelagem do treinamento usando Especificação de Sistemas de Eventos Discretos (DEVS) (quando o fator tempo implica em mudanças de estado) e/ou Autômato Determinístico Finito (DFA) (quando o tempo não influencia a simulação). Para a visualização e modelagem DEVS, podem ser usados: **CD++ Toolkit** [2011] desenvolvida em C++; **JAMES II** [2015] e **DEVSJAVA** [2015], ambas desenvolvidas em Java.

Para a modelagem DFA, podem ser usados: **Automata Editor** [2013]: editor para desenhar DFA, desenvolvido em C/C++; **JFLAP** [2015]: ferramenta gráfica de edição de autômatos formais determinísticos e não determinísticos (DFA e NFA), desenvolvida em Java; **Visual Automata Simulator** [2006]: ferramenta para simular, visualizar e editar DFA ou NFA, também desenvolvida em Java.

4.2. Ferramentas de Apoio à Implementação do Jogo Sérió

O jogo contém recursos (também chamados de *assets*) de texto, arte (imagens 2D e modelos 3D), áudio e vídeo, código etc. Após o projeto, este conteúdo deve ser criado ou reusado de repositórios (gratuitos ou comprados). Para criar, modelar e editar estes *assets*, diversas ferramentas podem ser usadas. A seguir são apresentadas as ferramentas de modelagem 3D, motores de jogos e banco de dados). A lista completa é relatada em [Rocha, 2015]. Elas devem ser instaladas e usadas em um computador local.

Para criação de modelo 3D, há *software* de modelagem paramétrica (sólidos 3D), tais como, o **BRL-CAD** [2013] desenvolvido em C, C++ e Tcl, e **FreeCAD** [2015] desenvolvido em C++ e Python; e *software* de modelagem gráfica que possui diferentes funções, que vão desde texturização até animação, tais como, **Blender** [2015] que possui até um motor de jogos, desenvolvido em C, C++ e Python, **Art of Illusion** [2015] modelagem e animação, desenvolvido em Java, e **Wings3D** [2014] inclui texturização, desenvolvido em Erlang.

Motores de jogo, em inglês *game engines*, são aplicativos e conjuntos de bibliotecas que abstraem funcionalidades importantes na criação de jogos e simulações [Feijó, Clua e Silva, 2010]. Essas funcionalidades podem ir desde a comunicação com dispositivos de entrada (teclado, *mouse*, *joystick*) e saída (*hardware* gráfico e de som), até cálculos físicos e inteligência artificial. Entretanto, a criação de um motor de jogo é uma tarefa complexa de programação e custosa em termos de tempo e dinheiro. Assim, empresas se especializaram em construir motores para jogos, ou componentes de motores. Assim, o desafio é escolher o motor que melhor atende os requisitos do projeto de jogo sério que será desenvolvido. Abaixo são listados motores de código livre e multiplataformas (Windows, Linux e MacOS). Todos estes motores possuem suporte à renderização, texturização, animação, simulação de física e malhas. Apenas o jME e o jPCT-AE foram desenvolvidos em Java, todos os outros usaram C/C++. **Blender Game Engine** (2015) é um sistema de modelagem 3D, animação, composição, renderização e desenvolvimento de simulações e jogos 3D em tempo real. **Crystal Space** (2014) é um *framework* para desenvolvimento de aplicações gráficas 3D, composto por um motor de renderização e um sistema de entidade, que suporta declarações específicas de funções em alto nível. **Delta3D** (2015) é um motor de simulação e jogo, que contém uma camada de APIs que integra diversas bibliotecas existentes para diferentes fins. Ele possui também um gerenciador de jogo que administra a comunicação entre as entidades estáticas e dinâmicas, e os dispositivos de entradas. **jMonkey Engine 3.0** (2015) é um motor desenvolvido em Java, baseado na API gráfica OpenGL. **jPCT-AE**

(2015) é um motor de renderização 3D orientado ao desenvolvimento de jogos para Java e Android. **OGRE** (2015) é um motor gráfico orientado a objetos. **OpenSceneGraph** é um *toolkit* gráfico de alto nível para o desenvolvimento de aplicações gráficas de alta performance, que possui um *framework* orientado a objetos, em cima da OpenGL, para o rápido desenvolvimento de aplicações gráficas. **Panda3D** (2015) é motor de jogo e um *framework* para renderização 3D, que utiliza um grafo de cena para gerenciar os objetos no ambiente virtual.

Um requisito importante dos jogos sérios é a avaliação do desempenho do aprendiz após o treinamento, por meio de relatórios. Para que isto aconteça, é necessário armazenar todos os dados do treinamento em banco de dados. Desta forma, é importante definir o Sistema de Gerenciamento de Banco de Dados (SGBD), relacional ou não relacional, e projetá-lo. Os principais SGBDs objeto relacional e multiplataforma são: **MySQL** (2015): mais comum, ideal para grandes volumes de dados, possui interface para diversas linguagens, tais como, C/C++, Java, Python e PHP; **PostgreSQL** (2015): de rápido desempenho, ideal para grandes volumes de dados, possui recursos de dados estatísticos e interface de programação para C/C++, Java, .Net, Python, Perl, Tcl, entre outras; e **Firebird** (2015): ideal para pequenos volumes de dados, possui componentes de acesso em C++, Java, .Net, Python, PHP, Perl etc.

5. Resultados e Discussões

Como resultado deste trabalho, foi feita uma revisão e seleção das principais ferramentas *open source* que podem ser utilizadas em conjunto com a metodologia de desenvolvimento de jogos sérios, para criar os artefatos em cada processo. Cada ferramenta pode ser melhor utilizada em um tipo de cenário de aplicação, de acordo com as necessidades. Dois exemplos de cenários desenvolvidos usando a metodologia criada são descritos a seguir. O primeiro é uma simulação simples 2D/3D, para celulares com o sistema operacional *Android*, para controle do painel da viatura do Corpo de Bombeiros (operação da bomba de água), conforme Figura 3 [Lima e Araujo, 2011]. Foram utilizadas as ferramentas Dia (UML), JFLAP (DFA), Blender, jPCT-AE (motor de jogos com programação em Java). O segundo exemplo é um jogo sério, com três fases 2D e três fases 3D, multiplataforma, com acesso *on-line* via navegador *Web*, para controle de vazamento de gás de cozinha, conforme Figura 4 [Rocha, 2014]. Neste caso, foram utilizadas as ferramentas *open source* ArgoUML (UML), Automata Editor (DFA), e o banco de dados foi desenvolvido em linguagem MySQL com acesso implementado em PHP. Já como motor de jogos, foi utilizado a versão gratuita da Unity3D© [2015], com programação na linguagem C#.

Figura 3. Simulação criada para celulares *Android* e seu DFA modelado usando JFLAP [adotado de Lima e Araujo, 2011]

Entender as características das plataformas e ferramentas é o primeiro passo tanto para poder utilizá-las, como para estendê-las, de modo a criar um ambiente e *framework* que possibilite mais recursos de apoio à equipe de desenvolvimento, tanto

nos aspectos de reuso e interoperabilidade de artefatos, quanto em sua geração automática. Como exemplos, as ferramentas de modelagem DFA e DEVS podem ser estendidas para gerar os modelos padronizados que servem de entradas para os componentes que simulam o treinamento e avaliação. Da mesma forma, as ferramentas de diagrama UML podem ser estendidas para contemplar as alterações realizadas no diagrama estruturado de classe, o qual especifica as características do objeto simulado.

Figura 4. Jogo sério multiplataforma acessado via Web [adotado de Rocha, 2014]

Além disso, a identificação dos formatos padrões, que as ferramentas geram para os diferentes *assets*, possibilitará a criação de um repositório compartilhado para o desenvolvimento de novos jogos sérios, podendo até, dada a padronização ou com uso de conversores, serem reusados em diferentes plataformas. De forma geral, ter um conjunto de opções de ferramentas possibilita a escolha da qual mais se adequa a cada cenário, além de abranger um número maior de profissionais.

6. Considerações Finais

Há uma lacuna entre as áreas que oferecem apoio à criação de jogos sérios (*design* de jogos, modelagem de simulações, *design* de instrução e treinamento), de modo que as metodologias propostas para o desenvolvimento de jogos sérios e educacionais são limitadas ao não abrangerem todos os requisitos necessários de cada área. Além disto, ao mesmo tempo que a metodologia tem que guiar e apoiar o desenvolvimento do jogo sério, de forma sistemática e padronizada, ela também tem que ser flexível para atender às diferentes necessidades dos profissionais que a estão usando. Desta forma, os produtos gerados e ferramentas usadas mudam dependendo das características do jogo sério e da equipe (quantidade e competências dos profissionais envolvidos).

Neste artigo, foi apresentado um conjunto de ferramentas *open source* que pode ser utilizado para criar os artefatos em cada processo da metodologia. Como trabalhos futuros, estas ferramentas deverão ser analisadas e estendidas, de modo a facilitar o trabalho dos profissionais envolvidos, por meio da geração de artefato, e seu reuso e interoperabilidade. Outros aspectos da metodologia também deverão ser aprofundados, tais como, o gerenciamento e a gestão da qualidade dos projetos e equipe de desenvolvimento (abordando as competências necessárias para cada ator); a identificação de cenários com diferentes necessidades de projeto do jogo sério para realização de provas de conceitos usando a metodologia e as ferramentas descritas; a definição de métricas e indicadores para avaliar o uso da metodologia.

Agradecimentos

Agradecemos ao CNPq, FAPESP e CAPES (projeto INCT-Sec e processos 2010/17005-7, 237216/2012-4 e 150613/2015-6), pelo apoio financeiro; ao Corpo de

Bombeiros do Estado de São Paulo, pelo auxílio; e a todos que participaram do projeto na Universidade Federal de São Carlos, pelas colaborações.

Referências

- ABNT. (2001). NBR ISO 10015:2001 - Gestão da Qualidade: diretrizes para treinamento. RJ: ABNT.
- ACUÑA, S.T.; FERRÉ, X. (2001). Software Process Modelling. In: Proceedings World Multiconference on Systemics, Cybernetics and Informatics: Information Systems Development. p. 237-242.
- ADAMS, W.K.; et al. (2007). A Study of Educational Simulations Part I – Engagement and Learning. *Journal of Interactive Learning Research*, v. 12. p. 397-419.
- AKILLI, G.K.; CAGILTAY, K. (2006). An Instructional Design/Development Model for the Creation of Game-Like Learning Environments: the FIDGE model. In: Pivec, M. (Ed.), *Affective and Emotional Aspects of Human-Computer Interaction: Game-Based and Innovative Learning*. IOS Press, p.93-112.
- ARGO (2009). Disponível em: <http://argouml.tigris.org>. Acesso em: Maio, 2015.
- ART OF ILLUSION (2015). Disponível em: <http://www.artofillusion.org>. Acesso em: Maio, 2015.
- ASLAN, S.; BALCI, O. (2015). GAMED: Digital Educational Game Development Methodology, In: *Simulation*, v. 91, n. 4, p. 307-319.
- AUTOMATA EDITOR (2013). Disponível em: <http://automataeditor.sourceforge.net>. Acesso em: Maio, 2015.
- BALCI, O. (2012). A Life Cycle for Modeling and Simulation. In: *Simulation*, v. 88, n. 7, p. 870-883.
- BANKS, J.; et al. (2001). *Discrete-Event System Simulation*. 3rd. ed. New Jersey, USA: Prentice-Hall.
- BECKER, K.; PARKER, J. (2012). Serious Instructional Design: ID for digital simulations and games. In: P. RESTA (ED.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2012*. Chesapeake: AACE, p. 2480-2485.
- BILLE, W.; et al. (2004). Using Ontologies to Build Virtual Worlds for the Web. In: *IADIS International WWW/Internet 2004 Conference*, October 06-09, 2004, Madrid, Spain. p. 683-690.
- BLENDER (2015). Disponível em: www.blender.org. Acesso em: Maio, 2015.
- BLENDER GAME ENGINE (2015). Disponível em: <http://www.blender.org>. Acesso em: Maio, 2015.
- BRL-CAD (2013). Disponível em: www.brlcad.org. Acesso em: Maio, 2015.
- C++ TOOLKIT (2011). Disponível em: <http://cell-devs.sce.carleton.ca>. Acesso em: Maio, 2015.
- CAMPOS, J.P.; GUIMARÃES, J.P. (2008). Em busca da eficácia em treinamento: Norma ABNT NBR ISO 10015:2001 - gestão da qualidade – diretrizes para treinamento. 107 p.
- CASSANDRA (2015). <http://cassandra.apache.org>. Acesso em: Maio, 2015.
- CHANDLER, H.M. (2012). *Manual de Produção de Jogos Digitais*. 2. ed. Porto Alegre: Bookman. 478 p.
- CRYSTAL SPACE (2014). Disponível em: <http://www.crystalspace3d.org>. Acesso em: Maio, 2015.
- DELTA3D (2015). Disponível em: <http://www.delta3d.org>. Acesso em: Maio, 2015.
- DEVJAVA (2015). Disponível em: <http://acims.asu.edu/software/devsjava>. Acesso em: Maio, 2015.
- DIA (2013). Disponível em: <https://wiki.gnome.org/Apps/Dia>. Acesso em: Maio, 2015.
- DICK, W.; CAREY, L. CAREY, J.O. (2004). *The Systematic Design of Instruction*. Allyn & Bacon.
- FEIJÓ, B.; CLUA, E.W.; SILVA, F.S. (2010). *Introdução à Ciência da Computação com Jogos: aprendendo a programar com entretenimento*, Rio de Janeiro: Elsevier.
- FIREBIRD (2015). Disponível em: <http://www.firebirdsql.org>. Acesso em: Maio, 2015.
- FORD, F. (2004). The Euclid RTP 11.13 SE Development & Exploitation Process (SEDEP). *European Simulation Interoperability Workshops*, 04E-SIW- 037, Edinburgh, Scotland, p. 1-10.
- FREECAD (2015). Disponível em: <http://www.freecadweb.org>. Acesso em: Maio, 2015.
- FREITAS, S.; JARVIS, S. (2006). A Framework for Developing Serious Games to meet Learner Need. *Interservice/Industry Training, Simulation, and Education Conference*. p. 1-11.
- GIMP (2014). Disponível em: <http://www.gimp.org/>. Acesso em: Maio, 2015.
- HAYS, R.T. (2005). The Effectiveness of Instructional Games: a literature review and discussion. *Naval Air Warfare Center Training Systems Division (Technical Report 2005 – 004)*.

- HBASE (2015). <http://hbase.apache.org>. Acesso em: Maio, 2015.
- IEEE (2003). 1516.3-2003- IEEE Recommended Practice for High Level Architecture (HLA): federation development and execution process (FEDEP). New York: IEEE.
- IEEE (2010). 1730-2010- Distributed Simulation Engineering and Execution Process (DSEEP). Washington: IEEE.
- INKSCAPE (2015). Disponível em: <https://inkscape.org/pt/>. Acesso em: Maio, 2015.
- JAMES II (2015). JAVa-based Multipurpose Environment for Simulation II. Disponível em: www.jamesii.org. Acesso em: Maio, 2015.
- JFLAP (2015). Disponível em: <http://www.jflap.org/jflaptmp>. Acesso em: Maio, 2015.
- JMONKEY ENGINE (2015). Disponível em: <http://jmonkeyengine.org>. Acesso em: Maio, 2015.
- JPCT-AE (2015) Disponível em: <http://www.jpct.net/jpct-ae>. Acesso em: Maio, 2015.
- LIMA, R. H. P.; ARAUJO, R. B. *Interface de Visualização 3D em Dispositivos Móveis para Simulações de Treinamento*. Relatório Científico (Bolsa FAPESP de Iniciação Científica) - Departamento de Computação, Universidade Federal de São Carlos, São Carlos, SP, 2011.
- KEMP, J. (1977). Instructional Design: a plan for unit and course development. Belmont: Fearon-Pitman.
- KIRKLEY, S.; TOMBLIN, S.; KIRKLEY, J. (2005). Instructional Design Authoring Support for the Development of Serious Games and Mixed Reality Training. I/ITSEC. Bloomington.
- MONGODB (2015). <http://www.mongodb.org>. Acesso em: Maio, 2015.
- MYSQL (2015). Disponível em: <https://www.mysql.com>. Acesso em: Maio, 2015.
- NOVAK, J. (2010). Desenvolvimento de Games. Trad. Pedro Cesar de Conti. São Paulo: Cengage. 443 p.
- OGRE (2015). Object-Oriented Graphics Rendering Engine. Disponível em: <http://www.ogre3d.org>. Acesso em: Maio, 2015.
- OPENSCENEGAPH (2007). Disponível em: <http://trac.openscenegraph.org>. Acesso em: Maio, 2015.
- PAIVA, D. C. (2006). Modelagem e Simulação de Multidões Humanas em Situações da Vida Cotidiana usando Ontologias. 145p. Dissertação (Mestrado em Computação Aplicada), Unisinos, São Leopoldo.
- PANDA3D (2015). Disponível em: <https://www.panda3d.org>. Acesso em: Maio, 2015.
- POSTGRESQL (2015). Disponível em: <http://www.postgresql.org>. Acesso em: Maio, 2015.
- ROCHA, R.V. (2014). Metodologia iterativa e modelos integradores para desenvolvimento de jogos sérios de treinamento e avaliação de desempenho humano. Tese (Doutorado em Ciência da Computação) – Departamento de Computação, Universidade de São Carlos, São Carlos, SP.
- ROCHA, R.V. (2015). Metodologia de Desenvolvimento de Jogos Sérios: especificação de ferramentas *open source* de apoio ao desenvolvimento. Relatório Técnico. São Carlos, SP.
- RODRIGUES, H.F.; MACHADO, L.S.; VALENÇA, A.M. (2010). Definição e Aplicação de um Modelo de Processo para o Desenvolvimento de Serious Games na Área de Saúde. In: Proc. Congresso da Sociedade Brasileira de Computação - Workshop de Informática Médica, p. 1532-1541.
- SCHUYTEMA, P. (2008). Design de Games: uma abordagem prática. SP: Cengage Learning. 447 p.
- SWARTOUT, B.; et al. (1997). Toward Distributed Use of Large-Scale Ontologies. In: Proceedings of AAAI97 Spring Symposium Series Workshop on Ontological Engineering: AAAI Press. p. 138-148.
- TRYBUS, J. (2010). Game-Based Learning: What it is, Why it Works, and Where it's Going. NMI White Paper. New Media Institute, New York.
- UMLET (2015). Disponível em: <http://www.umlet.com>. Acesso em: Maio, 2015.
- UNITY (2015). Disponível em: <http://unity3d.com>. Acesso em: Maio, 2015.
- VAN DER ZEE, D.J.; HOLKENBORGB, B.; ROBINSON, S. (2012). Conceptual modeling for simulation-based serious gaming. In: Decision Support Systems, v. 54, p. 33-45.
- VISUAL AUTOMATA SIMULATOR (2006). Disponível em: <https://www.cs.usfca.edu/~jbovet/vas.html>. Acesso em: Maio, 2015.
- WINGS3D (2014). Disponível em: <http://www.wings3d.com>. Acesso em: Maio, 2015.
- ZIN, N.A.M.; JAAFAR, A.; YUE, W.S. (2009). Digital Game-Based Learning (DGBL) Model And Development Methodology For Teaching History. WSEAS Transactions On Computers, v. 2, n. 8, p.322-333.