

Métricas para Avaliação de Motores de Jogos Tridimensionais

Ítalo Mendes da S. Ribeiro
italo.ribeiro@gmail.com

Selan Rodrigues dos Santos
selan@dimap.ufrn.br

Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Campus Lagoa Nova, 59072-970, Natal/RN

Resumo

Existe uma variedade de motores de jogos tridimensionais (3D) disponíveis para diversos objetivos, como a criação de jogos, simulações, animações entre outros. Contudo, também existe uma grande necessidade de definição de mecanismos que permitam realizar comparações entre os motores, levando em consideração, por exemplo, parâmetros como qualidade de software, funcionalidades suportadas, facilidade de uso. Outro fator determinante que deve ser levado em consideração em uma eventual comparação de motores é a natureza da aplicação que se deseja desenvolver com o auxílio do motor. Neste artigo propomos um sistema de métricas para a avaliação e comparação de motores de jogos. A título de validação, aplicamos este sistema de métricas na avaliação de dois populares motores de jogo 3D.

Keywords: motores de jogo 3D, métricas, desenvolvimento de aplicações gráficas.

1 Introdução

Os primeiros jogos eletrônicos eram bem simples e possuíam uma programação monolítica e dependente da plataforma. O lançamento dos títulos *Doom* e *Quake* [id software 2009] foi um momento marcante na evolução dos jogos eletrônicos, por apresentar um maior realismo gráfico e sonoro, com ações mais realistas, se comparado aos títulos da época. Isso foi possível devido ao aumento de desempenho dos computadores em geral, e o aprimoramento de hardware especializado, como é o caso das placas de aceleração gráfica com processadores dedicados (GPUs).

Outra grande contribuição destes títulos foi a introdução do conceito de *motores de jogos 3D*. Os motores podem ser considerados como uma camada extra de abstração separando a camada de aplicação dos comandos em mais baixo nível e específicos, como renderização gráfica, inteligência artificial e conexão em rede. A principal vantagem desta nova abordagem é a possibilidade de reutilizar o mesmo motor para títulos similares, bastando modificar a camada da aplicação.

Essa evolução conjunta de software (com os motores) e de hardware (com o hardware dedicado de baixo custo) foi responsável pelo notável crescimento da indústria de desenvolvimento de jogos. Atualmente os motores são empregados principalmente na criação de jogos eletrônicos, simulações físicas, médicas, composição de alguns elementos de animações, passeios virtuais interativos, etc. [Lewis and Jacobson 2002]

Contudo, essa mesma evolução trouxe consigo alguns novos desafios, como por exemplo um progressivo aumento na complexidade de desenvolvimento de jogos. Isso se deve, por exemplo pela a adição de mais elementos aos jogos, como simulação de Física, comunicação de grande quantidade de dados entre milhares de jogadores em rede, construções de ambientes 3D massivos, segurança na transmissão dos dados, etc. [Rollings and Morris 2003] Para superar estes desafios, foram desenvolvidos um grande número de motores de jogos com funcionalidades diversas, o grande obstáculo, portanto, passa a ser definir que motor é adequado para uma determinada aplicação.

Muitas vezes a escolha de um motor é feita com base no seu

custo ou puramente na experiência pessoal prévia dos desenvolvedores, ou seja, sem critérios objetivos e bem definidos. Desta forma, a proposta deste trabalho é apresentar um sistema de métricas para auxiliar na tomada de decisão com relação a escolha do motor mais apropriado para uma determinada aplicação. Para tanto, o sistema de métricas leva em consideração as diversas funcionalidade dos motores, ponderado pela categoria de aplicação a ser desenvolvida.

2 Trabalhos relacionados

A utilização de métricas não é novidade. Por exemplo, em [Eves and Meehan 2008] são apresentadas algumas métricas com pontuações associadas. No entanto, a pontuação é definida por um analisador que está aplicando as métricas para um determinado projeto para a construção de uma aplicação 3D. Isso pode tornar o resultado imparcial e variável, pois outra pessoa pode aplicá-las no mesmo projeto e os resultados serem muito diferentes.

O trabalho apresentado em [Korva 2007] cita muitos pontos que devem ser considerados para a escolha de um motor de jogos 3D, mas não define uma pontuação. Já a métrica proposta por Threnholme e Smith [Trenholme and Smith 2008] possui poucos pontos a serem analisados, não abrangendo de forma satisfatória todas as funcionalidades que um motor de jogos possui e a comparação entre os motores é realizada principalmente a partir do uso do motor, tornando a comparação pouco precisa.

Assim, a criação de um modelo de comparação entre motores de jogos 3D que aborde os vários aspectos (gráficos, acessórios, suporte e de software) existentes nessas ferramentas é necessário para que uma análise completa seja feita. A utilização de uma pontuação para cada métrica é útil para uma medida mais precisa de cada motor de jogos. Além disso, a pontuação pré-definida considerando a facilidade que o quesito adiciona a criação de aplicações 3D, evita que o resultado seja influenciado pela opinião de cada avaliador, eliminando as diferenças entre os resultados obtidos.

3 Métricas Propostas

Alguns elementos devem ser levados em consideração para a comparação entre motores de jogos. As suas funcionalidades são os principais fatores a serem avaliados, podendo ser classificadas em cinco categorias: *gráficas*, *acessórias*, *suporte*, *software* e *aplicações*. As funcionalidades gráficas compreendem os recursos que o motor dispõe para a renderização do ambiente. Nesta categoria se enquadram elementos como a geração de terreno, *skybox*, efeitos especiais como neblina, fogo e *flare*, texturas suportadas, geração de sombras, iluminação, etc.

Dentre as funcionalidades acessórias podem ser citadas a simulação de Física, muito importante para aumentar o realismo e dinamismo do ambiente; a conexão em rede, que possibilita a realização de partidas online através da rede de computadores; e a inteligência artificial, responsável por controlar os NPCs (*non-player character*), por exemplo. As funcionalidade relacionadas à suporte correspondem à documentação, suporte e a atividade da comunidade de desenvolvedores e utilizadores, uma vez que isso pode contribuir para a solução de problemas com o uso do motor.

As linguagens suportadas e as plataformas de execução de um motor são outros fatores importantes de comparação. Um outro elemento a ser considerado é a estruturação do motor, que acaba determinando tanto a forma de construção dos projetos quanto a facilidade de aprendizagem dele. Essas funcionalidades compõem as métricas de software.

Do conjunto de métricas citadas, algumas são mais importantes que outras em alguns tipos de aplicações. Por exemplo, uma aplicação de simulação possui necessidades bem distintas das de um jogo com finalidade de entretenimento. Dessa maneira, algumas métricas devem ser ressaltadas em relação a outras, dependendo da aplicação em desenvolvimento.

As métricas de qualidade de software como portabilidade, eficiência e usabilidade estão relacionadas às métricas que serão apresentadas direta ou indiretamente, através das plataformas suportadas pelos motores (portabilidade), da documentação (usabilidade) e facilidade de criação dos elementos do ambiente (eficiência).

Cada critério da métrica recebe uma pontuação variando de [0;3], de acordo com o grau de dificuldades tecnológica para a existência do recurso no motor e a facilidade que ela introduz na criação de uma aplicação 3D. A categorização da pontuação é descrita a seguir:

- 0 - Inexistente:** recurso não presente no motor.
- 1 - Básica:** recurso essencial, que deve estar presente em praticamente todo motor.
- 2 - Recomendada:** recurso intermediário mas que aumenta a produtividade da aplicação e conduz a um melhor efeito final para aplicação; presente na maioria dos motores.
- 3 - Avançada:** recurso existente em poucos motores, normalmente associado a efeitos e características específicas, que diferenciam o motor dos demais; muito útil no desenvolvimento.

Por exemplo, a presença de algum tipo de gerente de cena recebe pontuação 1, enquanto que se o mesmo motor apresentar uma técnica de otimização de cenário, como *octree* ou paginação de terreno, receberia nota 2.

O cálculo total da pontuação de um motor é o somatório de todos os pontos obtidos nas diversas categorias das métricas. As métricas que são incorporadas em um motor na forma de complemento (*add-ons*) também entram na contagem dos pontos. A Tabela 1 contém todas as métricas e suas respectivas pontuações, localizada ao lado de cada métrica.

As métricas onde a pontuação difere da forma estabelecida no início desta seção são marcadas com um “*” na Tabela 1 e serão discutidas a seguir.

3.1 Formatos Suportados

A pontuação deste quesito é indicado de acordo com a quantidade de formatos de imagens, malhas e áudio suportados pelo motor: **(1)** - 1 à 5 formatos; **(2)** - 6 à 10 formatos; e **(3)** - acima de 10 formatos.

3.2 API Gráfica

Um motor de jogos é construído sobre uma API gráfica, que é um conjunto de funções para acessar os componentes gráficos do dispositivo. As duas principais APIs gráficas utilizadas são OpenGL e DirectX. A pontuação é de acordo com as APIs suportadas pelo motor: **(1)** - Somente OpenGL ou DirectX; e **(2)** - OpenGL e DirectX.

3.3 Status

De acordo com o estado de desenvolvimento a pontuação adicionada para a avaliação é a seguinte: **(1)** - **Inativa:** o motor não está mais sendo mantido ou atualizado; **(2)** -

Beta: o motor ainda está em um estágio de testes; e **(3)** - **Estável:** o motor atingiu um estado estável, e pode ser utilizada para desenvolvimento sem problemas.

3.4 Custo

O custo, em muitos casos, é um fator decisivo na escolha do motor. Existem muitos motores que apesar de possuírem ótimas ferramentas e muitas funcionalidades, têm custos elevados, tornando-os inacessíveis para um único usuário ou pequenas empresas de desenvolvimento.

Dessa maneira, motores gratuitos ou de baixo custos recebem as maiores pontuações. A pontuação atribuída ao motor nesta categoria depende da faixa de valores em que a sua licença se enquadra: **(1)** - acima de US\$10000; **(2)** - entre US\$3000 e US\$9999; **(3)** - entre US\$1000 e US\$2999; **(4)** - entre US\$1 e US\$999; e **(5)** - gratuita.

Para um mesmo motor a licença pode variar de gratuitas com restrição de uso para algumas funcionalidades, até o preço completo, onde todas as suas funcionalidades estão disponíveis. Portanto, o processo de pontuação deve considerar apenas as funcionalidades presentes no motor de acordo com a licença escolhida.

3.5 Sistema Operacional

A quantidade e os tipos de sistemas operacionais (SOs) para os quais um motor de jogos pode gerar uma aplicação são importantes para os planos de negócio de uma empresa. O suporte do motor a diversas plataformas permite a definição dos dispositivos para os quais a aplicação poderá ser distribuída. Este fato influencia diretamente a abrangência da quantidade de usuários e consumidores da aplicação.

Os tipos de SOs suportados foram organizados de acordo com os dispositivos suportados: *computadores* (Windows, Linux(Unix), MacOS); *consoles* (GameCube, Nintendo Wii, Nintendo DS, modelos Playstation, modelos Xbox); e *dispositivos móveis* (iPhone, smartphones, pocket pcs).

A pontuação do tipo de SO é atribuída da seguinte forma: **(1)** - suporta somente um dos tipos de sistemas operacionais; **(2)** - suporta dois tipos; e **(3)** - suporta os três tipos.

A pontuação relacionada a métrica de quantidade de SO é adicionada conforme o número de SOs suportados pelo motor.

3.6 Linguagem de Programação

A linguagem de programação é um fator importante para a escolha do motor de jogos. Dependendo do tipo de linguagem utilizada, pode-se gerar aplicações com desempenho variável.

Quanto mais linguagens um motor de jogos suportar, maiores são as possibilidades de aplicações que podem ser geradas. Assim a pontuação desse quesito reflete a quantidade de linguagens suportadas.

4 Análise por tipo de aplicação

De acordo com o tipo de aplicação que será desenvolvida com o motor de jogos, algumas métricas tem uma maior importância em relação as outras, na facilitação da criação da aplicação. Isso deve influenciar na escolha do motor. Cada métrica apontada nos tipos de aplicações deve ter 3 pontos adicionados aos já estabelecidos anteriormente. Por exemplo, se 1 ponto é normalmente atribuído à métrica *shadow mapping* e este critério é relevante para um certo tipo de aplica, a pontuação deve ser acrescida de mais 3 pontos, totalizando 4 para o quesito.

Tabela 1: Métricas e suas respectivas pontuações, onde as letras ao lado do nome do grupo de métricas são G(gráficas), A(acessórios), Sp(suporte) e Sf(software).

Gerente de cena (G)		Iluminação (G)		Efeitos Especiais (G)		Texturização (G)		Animação (G)		Malhas (G)	
Geral	1	<i>Lightmaps</i>	1	<i>Billboarding</i>	1	Básica	1	<i>Keyframes</i>	2	Leitura	1
BSP	1	Por-Vértice	1	<i>Lens flare</i>	2	Multitextura	1	Por esqueletos	2	Deformação	3
Portais	2	Por-Pixel	2	Sist. partículas	2	<i>Bump mapping</i>	2	<i>Blending</i>	2	Progressivas	3
<i>Octrees</i>	2	Anisotropia	2	Prof. de campo	2	Formatos	*	<i>Inv. kinematics</i>	3	Formatos	*
Pag. paisagem	3	Volumétrica	3	Névoa ou Céu	2			<i>Fwd. kinematics</i>	3		
		Radiosidade	3	Água ou Fogo	3			<i>Morphing</i>	3		
		BRDF	3	<i>Motion blur</i>	3			Facial	3		
				Climáticos	3						
Shaders (G)		Terreno (G)		Sombras (G)		API Gráfica (G)		Sist. Op. (Sf)		Linguagens (Sf)	
<i>Vertex shader</i>	2	Básico	1	<i>Shadow map</i>	1	OpenGL	*	Tipo	*	C/C++,	
<i>Pixel shader</i>	2	Deformável	3	<i>Shadow volume</i>	3	DirectX	*	Quantidade	*	Java, etc.	*
Gerais (G)		Rede (A)		Física (A)		Int. Artificial (A)		Som (A)		Suporte (Sp)	
GUI	1	Estruturação	1	Básico	1	<i>Pathfinding</i>	2	Básico	1	Exemplos	1
Entrada/saída	2	Cliente/servidor	2	Deteção colisão	1	Tomada decisão	2	3D	2	Fórum	2
<i>Scripting</i>	2	Ponto a ponto	2	Veículo	3	<i>Script</i>	2	<i>Streaming</i>	2	<i>Chat online</i>	2
Possui algum		Segurança	3	<i>Ragdoll</i>	3	Formação	3	Formatos	*	Documentação	2
tipo de editor	3			Terreno	3						
Estado desenv.	*			Fluído	3						
Custo	*										

4.1 Educacionais

As aplicações educacionais objetivam auxiliar no processo de aprendizagem dos seus usuários sobre um determinado assunto [Bellotti et al. 2009]. Normalmente este tipo de aplicação não necessita de muitos recursos computacionais para sua execução, pois pretende-se atingir o maior número de possível de usuários. A iluminação, sombras, terrenos e muitos outros efeitos especiais, adicionados por *shaders*, normalmente não são essenciais em projetos desse tipo. Além disso, não costumam utilizar recursos de comunicação em rede, nem componentes de física ou de inteligência artificial, embora isso não signifique que eles não sejam empregados em jogos educacionais específicos.

Assim, as métricas que devem ser acentuadas para aplicações educacionais são: (1) **Sombras:** *shadow volume*; (2) **Texturização:** multitextura; (3) **Animação:** *keyframes*, baseada em esqueletos, *blending*, facial; (4) **Efeitos especiais:** *billboarding*, sistemas de partículas; (5) **Gerais:** *scripting*, editor de partículas, interface gráfica com usuários (GUI); e (6) **Som:** som 3D.

4.2 Simuladores

Os simuladores procuram criar, o mais próximo possível da realidade, determinadas atividades em um ambiente 3D [Stang 2003]. São muito utilizados para estudos científicos realizando a análise dos resultados em várias situações que poderiam acontecer no mundo real, como por exemplo no treinamento de profissionais através dos simuladores de cirurgias [Marks et al. 2007]. Os jogos desse gênero costumam fazer grande sucesso no público adulto como o simulador de voo *Flight Simulator* [Microsoft 2009].

São aplicações que contém muitos detalhes em sua composição, utilizam muitos recursos computacionais e possuem ótima qualidade gráfica. Muitas das funcionalidades de um motor são importantes no seu desenvolvimento. (1) **Gerenciamento de cena:** particionamento binário do espaço (BSP), *octrees*, portais, paginação de paisagem; (2) **Iluminação:** anisotropia, volumétrica, radiosidade, BRD; (3) **Sombras:** volume de sombras; (4) **Texturização:** multitextura, *bump mapping*; (5) **Shaders:** *vertex* e *pixel shaders*; (6) **Malhas:** deformação; (7) **Efeitos especiais:** *lens flare*, sistema de partículas, profundidade de campo, céu, água, fogo, névoa; (8) **Terreno:** deformável;

(9) **Gerais:** GUI; editores de *script*, caminhos, terreno, partícula; (10) **Som:** som 3D e *streaming*; (11) **Física:** deteção de colisão, veículo, *ragdoll*, fluídos, corpos rígidos; e (12) **Inteligência artificial:** *pathfinding*

4.3 Visualização

Esse tipo de aplicação serve para a visualização de ambientes 3D, ou reconstruções tridimensionais a partir de imagens médicas bidimensionais. Podem também ser utilizadas para a visualização e manipulação de informações ou dados com muitas dimensões [Harrop and Armitage 2006].

Os visualizadores são aplicações em que o principal foco está na forma como será feita a visualização dos ambientes. Portanto, recursos como *shaders*, efeitos especiais, e inteligência artificial não são imprescindíveis para aplicações desse tipo. Assim métricas importantes são: (1) **Sombras:** volume de sombras; (2) **Texturização:** multitextura, *bump mapping*; (3) **Animação:** *keyframes*, baseada em esqueletos; (4) **Efeitos especiais:** *lens flare*, sistema de partículas, profundidade de campo, céu, água, fogo, névoa; (5) **Gerais:** interface gráfica com usuário e suporte a vários formatos de imagem; (6) **Som:** som 3D e *streaming*; (7) **Física:** deteção de colisão, veículo, *ragdoll*, fluídos, corpos rígidos; e (8) **Rede:** conexão cliente-servidor ou *multi-cast*.

5 Avaliações

Afim de demonstrar a utilização das métricas, elas foram utilizadas para a avaliação de dois motores de jogos populares.

5.1 OGRE

A OGRE (*Object-Oriented Graphics Rendering Engine*) é um motor voltado para a manipulação gráfica escrito em C++, não sendo um motor de jogos completo [Junker 2006]. Quando é necessária a utilização de outros recursos como áudio e física por exemplo, outras bibliotecas podem ser unidas ao OGRE através de seus *add-ons* que facilitam a integração. É um motor gratuito, de código aberto, multiplataforma e em pleno desenvolvimento. Possui versões em outras linguagens como Java e Python. A comunidade de usuários é grande e muito ativa.

A OGRE possui ótimos gráficos e muitos efeitos visuais, mas não possui bons editores. Os elementos na cena são organi-

zados em uma estrutura hierárquica de nós de cena. A documentação é bem completa e possui uma grande contribuição da comunidade, embora haja a necessidade de atualizações nos textos disponíveis.

A OGRE conseguiu 116 pontos dos 177 que poderia obter caso obtivesse pontuação máxima em todas as métricas, conseguindo assim 65,53% da pontuação total. Nos tipos de aplicações conseguiu 152 (116 + 32) pontos nas educacionais, onde obteve 32 dos 42 pontos que poderiam ser obtidos (85,71%). Nos simuladores 209 (116 + 93), com 70,45% dos 132 possíveis e 146 (116 + 30) nas de visualização, onde obteve 90,90% do máximo que é de 33 pontos.

5.2 Irrlicht

A Irrlicht é um motor escrito em C++ simples e rápido, muito bom para os iniciantes na área de desenvolvimento de jogos, pois com poucas linhas de código é possível construir uma aplicação [Gebhardt 2009]. Possui uma documentação básica, embora seja um bom começo para os novos usuários. É também gratuita e de código aberto.

A organização dos componentes da cena é semelhante a OGRE, utilizado nós de cena. A comunidade é bem ativa com muitas participações no fórum e no desenvolvimento de complementos para o motor. Possui um editor de ambiente 3D nativo, o *IrrEdit*, qual possui muitos dos elementos necessários para a construção da cena que podem ser trabalhados, como luzes, objetos e terreno. O motor possui um sistema de detecção de colisões simples, um editor de interface com usuário. As bibliotecas para facilitar a integração são a *IrrNet* (usa Enet) para conexão em rede, a *IrrAI* para inteligência artificial *IRrklang* para áudio.

A avaliação da Irrlicht resultou na pontuação 104 representando 58,75% do total de pontos que poderia obter. Para aplicações educacionais conseguiu 137 (106 + 33) pontos, aproveitamento de 75%, simulações 179 (106 + 75), com 56% do total e nas de visualização 131 (106 + 27), obtendo 81,81%.

5.3 Discussão dos resultados

Observando os resultados da OGRE e da Irrlicht, a primeira mostrou-se um melhor motor com uma pontuação um pouco superior a da segunda. Um dos principais diferenciais que aumentaram os pontos da OGRE em relação a Irrlicht foi a quantidade de efeitos visuais oferecidos por ela. Isso se deve devido a OGRE ser, em sua essência, um motor gráfico. Um outro fator determinante foram funcionalidades encontradas somente na OGRE, como a animação facial.

A Irrlicht é um motor mais voltado para o desenvolvimento de jogos, mais simples e prático de ser utilizado. Oferece um conjunto de editores que agilizam o processo de desenvolvimento e que juntamente com o suporte a inteligência artificial ajudaram no aumento da sua pontuação e são um diferencial em relação a OGRE.

Nas aplicações educacionais e de visualização os dois motores mostraram-se ótimas opções, com pontuações muito próximas e boas (152 e 137, 146 e 131). Para aplicações de simulação a OGRE mostrou-se melhor devido aos efeitos visuais oferecidos, pois aumenta o realismo visual de uma simulação.

6 Conclusão e trabalhos futuros

A avaliação de motores ainda é um processo em desenvolvimento. Assim, as métricas propostas procuram criar uma referência inicial para a comparação entre os motores de jogos 3D, que são utilizados para os mais diversos tipos de aplicações 3D. De acordo com a aplicação a ser desenvolvida, algumas métricas tem uma maior importância em relação

a outras, e mesmo assim ajustes são necessário para casos específicos.

O processo de avaliação procura analisar os motores de forma precisa e padronizada, sem muitas variações como ocorre na maioria das comparações realizadas em trabalhos anteriores, nos quais normalmente a análise é baseada na experiência do uso do motor pelo autor. Como proposto, não importa quem realize a avaliação a pontuação da OGRE e da Irrlicht serão as mesmas ou muito próximas (claro, assumindo que os pesos dos critérios são os mesmos).

Com a natural evolução das motores de jogos novas métricas podem ser adicionadas, e outras terem sua pontuação modificada. Novas plataformas e tipos de aplicações também devem serem consideradas e incluídas nas métricas, acomodando critérios específicos. Por exemplo, os jogos para celulares possuem características particulares, onde fatores como consumo de energia, uso restrito de memória e poder de renderização limitados devem ser considerados.

O próximo passo será o aperfeiçoamento das métricas propostas, aplicando-as a diversos motores, para então validar os resultados obtidos.

Referências

- AUTODESK, 2009. 3d studio max. <http://www.autodesk.com/3dsmax>, acessado em maio 2009.
- BELLOTTI, F., BERTA, R., GLORIA, A. D., AND PRIMAVERA, L. 2009. Enhancing the educational value of video games. *Comput. Entertain.* 7, 2, 1–18.
- EVES, G., AND MEEHAN, P., 2008. The development of selection criteria for game engines in the development of simulation training systems.
- GEBHARDT, N., 2009. Irrlicht engine. <http://irrlicht.sourceforge.net/>, acessado em junho 2009.
- HARROP, W., AND ARMITAGE, G. 2006. Modifying first person shooter games to perform real time network monitoring and control tasks. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, ACM, 10.
- ID SOFTWARE, 2009. Quake. <http://www.idsoftware.com/games/quake/quake/>, acessado em maio 2009.
- JUNKER, G. 2006. *Pro OGRE 3D Programming*. Apress.
- KORVA, T., 2007. Open source & low cost game engines.
- LEWIS, M., AND JACOBSON, J. 2002. Game engines in scientific research - introduction. *Communications of the ACM* 45, 1, 27–31.
- MARKS, S., WINDSOR, J., AND WÜNSCHE, B. 2007. Evaluation of game engines for simulated surgical training. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, ACM, New York, NY, USA, 273–280.
- MICROSOFT, 2009. Microsoft flight simulator. <http://www.fsinsider.com/>, acessado em maio de 2009.
- ROLLINGS, A., AND MORRIS, D. 2003. *Game Architecture and Design*. New Riders, Pearson Education.
- STANG, B. 2003. Game engines: Features and possibilities. Tech. rep., Institute of Informatics and Mathematical Modeling at The Technical University of Denmark.
- TRENHOLME, D., AND SMITH, S. P., 2008. Computer game engines for developing first-person virtual environments.