UM ESTUDO COMPARATIVO DE FERRAMENTAS PARA A CRIAÇÃO DE JOGOS EDUCACIONAIS BASEADOS EM REALIDADE VIRTUAL

José Cavalcanti de Moura Netto[1], Liliane dos Santos Machado[1] e Ronei Marcos de Moraes[2]

[1]Departamento de Informática e [2]Departamento de Estatística
Universidade Federal da Paraíba
Cidade Universitária s/n – João Pessoa/PB - Brasil

e-mail: *netto.cavalcanti@gmail.com, liliane@di.ufpb.br, ronei@de.ufpb.br*

Resumo – *O aumento da demanda de jogos em todo o mundo mostra uma crescente necessidade de mão-de-obra qualificada. Neste contexto, as estatísticas para o mercado de jogos eletrônicos no Brasil e no mundo são promissoras. Os jogos eletrônicos, devido à sua complexidade, requerem técnicas e tecnologias recentes, tornando-os aplicações ricas em novos conhecimentos. Derivados dos jogos eletrônicos, os jogos eletrônicos educacionais possuem o mesmo potencial, acrescido de conhecimentos didáticos ou educativos. Este artigo apresenta um estudo comparativo de ferramentas para o desenvolvimento de jogos eletrônicos, explicitando suas potencialidades em relação ao suporte a técnicas e dispositivos de realidade virtual no desenvolvimento de aplicações.*

Palavras-Chave – Jogos Educacionais, Jogos eletrônicos, Motores de jogo, Realidade Virtual.

Abstract – *The increase of the demand of games in the world requires qualified professionals. In this context, are promising the surveys about market of electronic games in Brazil and in the world. The electronic games, due to its complexity, require recent techniques and technologies, what make the games rich applications of innovative knowledge. Educational electronic games have the same potential of the electronic games, added by didactic or educational knowledge. This article presents a comparative study of tools for the development of electronic games, showing their potentialities in relation to the support of techniques and devices in the development of games based on virtual reality.*

Keyword – Educational Games, Electronic Games, Game Engine, Virtual Reality.

1. INTRODUÇÃO

Com o constante aumento do mercado de jogos surgiu a necessidade de mão-de-obra qualificada, com conhecimentos mais voltados à teoria e conceituação do que à produção de jogos [9]. Ao desenvolver um jogo é necessário levar em conta quais passos devem ser seguidos desde a concepção e documentação da idéia, desenho das imagens do jogo, escolha de ferramentas, escolha das linguagens de programação e produção do jogo. Esta seqüência de passos é fundamental para a criação de aplicações robustas, de forma que o produto final não deixe de oferecer atração ou diversão para o usuário final, comprometendo seu valor ou simplesmente não atingindo o mercado visado.

Jogos eletrônicos (JE) são aplicativos com alto grau de complexidade e exigem o uso de técnicas diversas da computação como: linguagens de programação, sistemas operacionais, computação gráfica, inteligência artificial e sistemas de redes, além de envolverem diversas outras áreas como psicologia, pedagogia e artes, dentre outras [1]. Como extensão, os jogos eletrônicos educacionais (JED) são também aplicações do tipo *Computer Assisted Instruction* utilizadas para divertir os usuários, aumentando as chances de aprendizado dos conceitos, conteúdo ou habilidades embutidas no jogo [2]. As principais diferenças entre os JEs e os JEDs são: enredo e o foco. Assim, enquanto o desenvolvimento de JEs busca um equilíbrio entre violência e efeitos gráficos realistas, nos JEDs procura-se equilibrar o contexto educacional à diversão [1].

Para o desenvolvimento de JEDs, técnicas de Realidade Virtual (RV) podem ser utilizadas como atrativo para aumentar os níveis de imersão e interatividade do usuário, despertando o interesse do usuário. Assim, poderiam ser utilizados, por exemplo, dispositivos específicos para a estereoscopia como: vídeo-capacetes, dispositivos para isolamento visual entre o usuário e o mundo real; óculos obturadores, que filtram as duplas imagens geradas alternadamente pelo computador afim de sincronizá-las para a sensação de imagens com profundidade; óculos colorido azul e vermelho para visualização de imagens por anaglifo; e óculos polarizados utilizados em *Caves* e muros de visualização, que utilizam a estereoscopia de forma análoga aos óculos coloridos, porém com imagens em cores reais.

Pode-se definir uma aplicação baseada em RV de três formas: Passiva, Ativa, Interativa. Na sessão Passiva permite-se a exploração de um ambiente sem interação ou interferência, atuando como observador de um cenário sem poder modificar os caminhos e os pontos de visualização que são definidos por software. Durante uma sessão Exploratória o usuário pode controlar os caminhos seguidos, tornando a exploração do ambiente dirigida pelo usuário, mas não tem o controle sobre a interação com nenhum outro objeto da cena. Por outro lado, a sessão Interativa é a mais completa de todas as sessões, pois o usuário poderá observar, controlar os caminhos e pontos de

observação, além de poder interagir com objetos da cena e esses objetos responderão e reagirão às interações do usuário [5].

Este artigo apresenta um estudo comparativo de ferramentas para o desenvolvimento de jogos eletrônicos, explicitando suas potencialidades em relação ao suporte a técnicas e dispositivos de realidade. Com isto, pretende-se relacionar as ferramentas estudadas ao desenvolvimento de um jogo eletrônico educacional baseado em RV.

2. DESENVOLVENDO UM JOGO

Desenvolver um jogo envolve uma série de passos, que podem ser divididos em [1]:

1. concepção e documentação da idéia do jogo;
2. enredo e cronologia;
3. criação e descrição de personagens e itens;
4. estudo de ferramentas e
5. implementação do jogo.

O primeiro passo - concepção e documentação da idéia do jogo - consiste da ordenação das idéias do jogo concebido e sua documentação. Este passo objetiva não perder o foco principal da idéia, assim como documentar como pretende-se desenvolver esta idéia, quais as necessidades específicas do jogo e o seu público alvo. Finalizado o passo anterior, deve-se prosseguir com o enredo do jogo ou a história do jogo. Ou seja, deve-se definir quais caminhos poderão ser tomados pelo(s) personagem(s), suas falas e quais as entidades do jogo. Esta etapa pode ser considerada como a cronologia do jogo, pois ela define a ordem de ocorrência dos eventos. O terceiro passo é a criação e definição dos personagens e itens, ou seja, definir qual é o comportamento dos personagens, suas aparências, qualidades e defeitos. Junto a estas biografias deve ser produzido um desenho de cada personagem ou entidade que necessite ter uma presença visual no jogo, como vendedores do mercado, monstros, veículos, entre outros. Os itens também deverão ter sua descrição e desenho. Se estes passos foram seguidos cuidadosamente, ter-se-á uma documentação de todas as etapas anteriores esta síntese de documentos cria um artefato geralmente denominado *Design Bible*, que se assemelha a *Story Board* utilizada no cinema. A *Design Bible* será a fonte de todos os dados utilizados para a produção do jogo.

O quarto passo para produção de um JE, é o estudo e análise das ferramentas e linguagens de programação a serem utilizadas, as ferramentas para a produção de imagens e desenhos tridimensionais, ferramentas de áudio para a composição e edição dos efeitos sonoros, e outras ferramentas utilizadas em redes e vídeos. Ao estudar uma linguagem específica leva-se em consideração seus benefícios, como velocidade da aplicação, se há suporte para redes, suporte à aceleração gráfica, suporte à áudio estéreo, dentre outros. Das várias linguagens utilizadas hoje em dia como JAVA, .Net e C#, uma linguagem amplamente utilizada na produção de qualquer aplicação e também em jogos eletrônicos é C/C++. Especificamente para este trabalho, as linguagens de programação devem possuir características que permitam a utilização de dispositivos de RV. Levando em consideração que qualquer linguagem que tenha acesso de baixo nível pode implementar códigos para a utilização de tais dispositivos, observa-se que linguagens como JAVA e C/C++ permitem esse acesso. Desse modo, estas linguagens podem trabalhar diretamente com o *hardware*, programando-o para operar em aplicações específicas. Este é o caso de aplicações que trabalham com estereoscopia, que necessitam apenas de um *hardware* apropriado e um *software* que interprete suas configurações para funcionar adequadamente.

Para a implementação do jogo faz-se necessário o uso de uma ferramenta de auxílio geralmente denominada motor do jogo, um conjunto de bibliotecas e funções que facilitam a sua implementação. Quando são necessárias características específicas não encontradas nos motores disponíveis, é necessário criar um motor para o jogo. Atualmente, existem vários motores de jogos, tanto privados, como o motor do jogo *Half-Life 2* da Valve [8], quanto livres, como o Ogre3D [6].

Para este trabalho serão abordados os motores de jogo Ogre3D [6] e o Panda3D [7]. Dentre outros motores de acesso livre, como o Fly3D [4] e CrystalSpace3D [3], o Ogre3D e o Panda3D oferecem suporte ao uso de RV.

2.1. Ogre3D

O pacote de desenvolvimento Ogre3D [6] é um conjunto de bibliotecas escritas em C++ de código aberto utilizadas como um motor de renderização gráfica 3D em tempo real. Com este conjunto de bibliotecas é possível desenvolver jogos, visualizações de arquitetura, simulações ou qualquer aplicação que necessite de uma solução para renderização gráfica.

Muito utilizada no desenvolvimento de jogos, o Ogre3D possui características para facilitar o desenvolvimento de uma aplicação gráfica, já que oferece recursos que aumentam o desempenho durante a renderização de cenas e objetos tridimensionais.

Além de possuir alguns outros conjuntos de bibliotecas que visam facilitar a produção de aplicações, como bibliotecas de manipulação sonora, tratamento de entrada de dados por teclado e mouse, o Ogre3D trabalha em conjunto com um motor próprio para cálculo e execução de propriedades físicas para objetos e cenas – ODE (*Open Dynamic Engine*) – o que facilita ainda mais a produção de um jogo eletrônico.

O Ogre3D também possui recentes técnicas e tecnologias na área de computação gráfica como sombreamento por vértice e por fragmentos de programas; multi-textura; animação independente de grama (ou pêlos); suporte a arquivos comprimidos (arquivos Zip e PK3); sistema de partículas; efeitos especiais com transparência.

Quanto à RV, o Ogre3D dá suporte às mais variadas formas de visualização estereoscópica (anaglifo, luz polarizada), *Head Mounted Displays* (HMDs), *Caves* e muros de visualização.

Segundo experimentos, [6] para se trabalhar bem com estereoscopia junto à Ogre3D é necessário a instalação do pacote *ForceWare* da NVIDIA e a aplicação *3DStereo*, ambos disponíveis apenas para o sistema operacional Windows. Apesar da falta de suporte a aplicações estereoscópicas baseadas em sistemas operacionais livres, o Ogre3D é de uso livre e largamente utilizado pela comunidade desenvolvedora de jogos. Tal fato faz com que possua uma grande comunidade de usuários que colaboram com sua melhoria.

2.2. Panda3D

Criada com a linguagem C++, o Panda3D [7] é um conjunto de bibliotecas C++ que utiliza a linguagem *script* Python para tratar estas bibliotecas, elevando o nível de abstração durante a programação do jogo. Tal característica permite obter uma curva de aprendizado alta em pouco tempo.

O Panda3D utiliza o conceito de objeto e nodos, otimizando o processo de renderização da cena, já que, dado um nodo com vários filhos, estes filhos terão os mesmo atributos que o nodo pai como cor, iluminação, transparência e movimento, dentre outras características. Este motor possui também vários outros aspectos importantes para a criação de jogos eletrônicos como diversos efeitos de sombreamento (desenho, *deferred*, etc.), efeitos de *motion trails*, dupla visão, distorção tipo *wings of blue* e efeito redemoinho, dentre outros. Como características, o Panda 3D oferece efeitos de iluminação, efeito de fumaça, detecção de colisão, sistema de partículas, e animação por texturas.

Como destaque principal, o Panda3D tem a sua concepção e desenvolvimento voltados para aplicações envolvendo RV e computação gráfica. Desse modo, oferece suporte a dispositivos específicos de RV e tem uma adaptação simples para as várias formas de projeção estereoscópica. Neste último caso, uma simples modificação na linha de comando do arquivo de configuração muda o modo de projeção de simples para anaglifo. Caso o *hardware* esteja devidamente instalado e configurado, uma modificação equivalente pode ser feita no mesmo arquivo mudando a projeção para estéreo obturado ou polarizado.

3. COMPARATIVO DAS BIBLIOTECAS

Possuindo arquiteturas muito parecidas, já que ambas utilizam o conceito de gerenciamento por árvore e baseadas em linguagem C++, ambos os motores de jogos, Ogre3D e Panda3D, tem um potencial muito parecido para a criação de jogos eletrônicos baseados em RV.

Uma das maiores vantagens do Panda3D é sua curva de aprendizado, pois sua programação se trata de linguagem *script* que possui sintaxe mais simples que as linguagens de programação orientada a objeto, como C++. No caso de tratamentos bidimensionais, o Panda3D possui uma parte de seu sistema gráfico responsável pelo tratamento da interface bidimensional. Já no Ogre3D é necessário instalar a interface bidimensional CEGUI e suas dependências para poder tratar as interfaces do jogo.

Quanto à compatibilidade, o Ogre3D a possui não só com a OpenGL, mas também com o DirectX, aumentando sua distribuição perante os usuários, principalmente do sistema operacional *Windows*. Isto possibilita uma maior extensibilidade, já que se pode programar tanto na linguagem OpenGL quanto na DirectX. Já o Panda3D só tem compatibilidade com a linguagem OpenGL. Como o Ogre3D só utiliza a linguagem de programação C++ torna-se mais fácil fazer uma extensão de sua *Aplication Program Interface* (API), pois no Panda3D é necessário criar as classes da extensão e os *scripts* em Python que façam a comunicação entre o motor do Panda3D e as novas classes.

Ambos os motores são multiplataforma, abertos e de uso livre, porém o Ogre3D abrange uma maior número de sistemas operacionais, o que torna maior a sua comunidade de programação, com mais contribuições, e de usuários finais das aplicações por ele geradas. A Tabela I apresenta um quadro comparativo entre o Ogre3D e o Panda3D.

TABELA I – Características dos motores gráficos para jogos Ogre3D e Panda3D.

Comparativo entre os motores de jogo	
Ogre3D	**Panda3D**
Código aberto	Código aberto
Baseado em Linguagem C++	Baseado em Linguagem C++
Documentação de toda sua arquitetura	Documentação de toda sua arquitetura
Código orientado a objeto com gerenciador de cena que gerencia nodos	Código orientado a objeto com gerenciador de nodos
Compatibilidade com OpenGL e DirectX	Compatibildade com OpenGL
Multiplataforma – Windows, Linux, MacOS	Multiplataforma – Windows, Linux
Programação em C++	Programação em Python
Extensão apenas em C++	Extensão em C++ e Python
Interface tratada por biblioteca incorporada - CEGUI	Interface própria do Panda3D
Permite características de RV	Permite características de RV

4. CONCLUSÕES

Este artigo apresentou um estudo sobre os motores Panda3D e Ogre3D e uma análise comparativa de suas funcionalidades quando utilizados recursos de RV em aplicações com estes desenvolvidas. Neste caso, observou-se que ambas apresentam código aberto e são de uso livre, bem como oferecem suporte a interação e visualização estereoscópica.

O Ogre3D oferece uma maior portabilidade por ser utilizável em vários sistemas operacionais como Windows, Linux e MacOS, além de utilizar outra linguagem gráfica no auxílio de programação. No entanto, para quem deseja um motor de fácil aprendizado, o Panda3D fornece uma curva de aprendizado mais eficiente, já que a programação de aplicações no Panda3D é feita a partir de linguagem *script*.

No caso de aplicações de RV, o Panda3D é mais fácil de configurar e tem suporte para configuração no sistema operacional Linux. O Ogre3D também oferece suporte às aplicações de RV, mas possui uma comunidade muito voltada ao suporte para o sistema operacional Windows, o que dificulta sua utilização entre os que trabalham com *software* livre.

Os resultados deste trabalho serão aplicados no desenvolvimento de um jogo eletrônico educacional baseado em RV para ensino de geografia voltado para estudantes de 4ª e 5ª serie. Neste aspecto, a utilização do motor Panda3D mostrou-se mais eficaz que o Ogre3D devido à sua praticidade no desenvolvimento de um jogo e o suporte a dispositivos e técnicas de RV. Sob o ponto de vista do desenvolvimento de aplicações educacionais livres, o Panda3D é muito favorável por permitir sua utilização com suporte à RV em sistemas operacionais abertos como o Linux.

AGRADECIMENTOS

Este trabalho é parcialmente financiado pela FINEP através do projeto "EDUGAMES - Jogos educacionais para ensino de componentes matemáticas e geográficas usando realidade virtual e telefones celulares com software livre" e pelo CNPq através de bolsa institucional PIBIC/UFPB.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BATTAIOLA, André L. Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação. In: XIX Jornada de Atualização em Informática. Curitiba: SBC, julho/2000, v. 2, pp. 83-122.

[2] COBURN, Peter; Kelman, Peter; Roberts, Nancy et al. Informática na educação**.** Rio de Janeiro; LTC, 1988.

[3] CRYSTAL Space3D. Online: www.crystalspace3d.org – Acessado em: 18 de outubro de 2006.

[4] FLY. Online: www.fly3d.com.br - Acessado em: 18 de outubro de 2006.

[5] NETTO, Antônio Valério; Machado, Liliane dos Santos; Oliveira, Maria Cristina F. Realidade Virtual - Fundamentos e Aplicações. Visual Books, 2002.

[6] OGRE3D. Open Graphics Rendering Engine 3D. Online: www.ogre3d.org - Acessado em: 11 de maio de 2006.

[7] PANDA3D. Online: www.panda3d.org – Acessado em: 21 de junho de 2006.

[8] VALVE. Online: http://www.valvesoftware.com/ - Acessado em: 18 de outubro de 2006.

[9] ZYDA, Michael. Educating the Next Generation of Game Developers. Computer, IEEE, 39(6), 2006, pp. 30-34.