Framework para Desenvolvimento de Jogos 3D Baseado na API O3D

Leandro Bezerra Rodrigues
Ciência da Computação, Centro Universitário Serra dos Órgãos (UNIFESO), Teresópolis-RJ, Brasil
bzrrale@gmail.com

Júlio César da Silva
Ciência da Computação, Centro Universitário Serra dos Órgãos (UNIFESO), Teresópolis-RJ, Brasil e Mestrado Profissional em Educação Matemática, Universidade Severino Sombra (USS), Vassouras-RJ, Brasil
jcesarop@gmail.com

Resumo: *Este trabalho apresenta um estudo sobre as atuais tecnologias disponíveis para exibição de conteúdo multimídia 3D na internet. Entre as aplicações-chave nesse mercado estão os jogos eletrônicos on-line. O foco do trabalho foi analisar as tecnologias para desenvolvimento de jogos e propor um framework baseado nessa análise, específico para o desenvolvimento de jogos 3D para internet, o que facilitará o trabalho de tarefas rotineiras do desenvolvedor. Este trabalho foi dividido em três etapas: 1) estudo das etapas de desenvolvimento de jogos 3D; 2) análise das tecnologias de gráficos 3D para internet; 3) desenvolvimento de uma proposta de framework baseado na API O3D de gráficos 3D para internet, a partir dos conhecimentos adquiridos nas etapas anteriores.*

Palavras-Chave: *Desenvolvimento de Jogos, API Gráfica O3D e Framework.*

Framework for Development of 3D Games Based on Api O3D

Abstract: *This paper presents a study on the current technologies available for displaying 3D multimedia content on the Internet. Among the key applications in this market are the games online. The focus of the study was to analyze the technologies for game development and propose a framework based on this analysis specific to the development of 3D games for the Internet, facilitating the work of routine tasks of the developer. This work can be divided into three stages: 1) study of the stages of development of 3D games, 2) analysis of 3D graphics technology for Internet, 3) development of a proposed*

framework based on the 3D graphics API O3D for Internet, based on the knowledge acquired in previous steps.

Keywords: *Game Development, Graphical API O3D and Framework.*

1. Introdução a API O3D

A internet é considerada uma plataforma rica em recursos, pois, entre as diversas aplicações que existem, especificamente, para funcionarem na internet, os jogos *on-line* ganham cada vez mais espaço, especialmente, os ditos 3D. Para isto, existem várias ferramentas que permitem a criação de gráficos 3D para a internet (Uol 2009), porém um número reduzido propõe englobar o desenvolvimento de jogos, que não são constituídos apenas de gráficos 3D, mas possuem diversos processos e necessidades que vão da lógica do jogo até simples efeitos sonoros (Santos, Battaiola e Dubiela 2004).

Atualmente, a *engine Unity3D* para criação de jogos 3D (Clua e Bittencourt 2005) é uma das mais difundidas no meio comercial. Essa *engine* é um ótimo exemplo para apresentar o potencial do mercado que movimenta essa área. A *Unity3D* possui licença grátis e uma vasta quantidade de ferramentas para auxiliar o desenvolvedor. Todavia, o Google lançou em 2009 uma API gráfica específica para a internet, chamada *Google* O3D. O foco desta API é fazer uma ponte entre o navegador e as bibliotecas *OpenGL* e *Direct3D*. A Figura 1 apresenta a arquitetura desta API O3D.

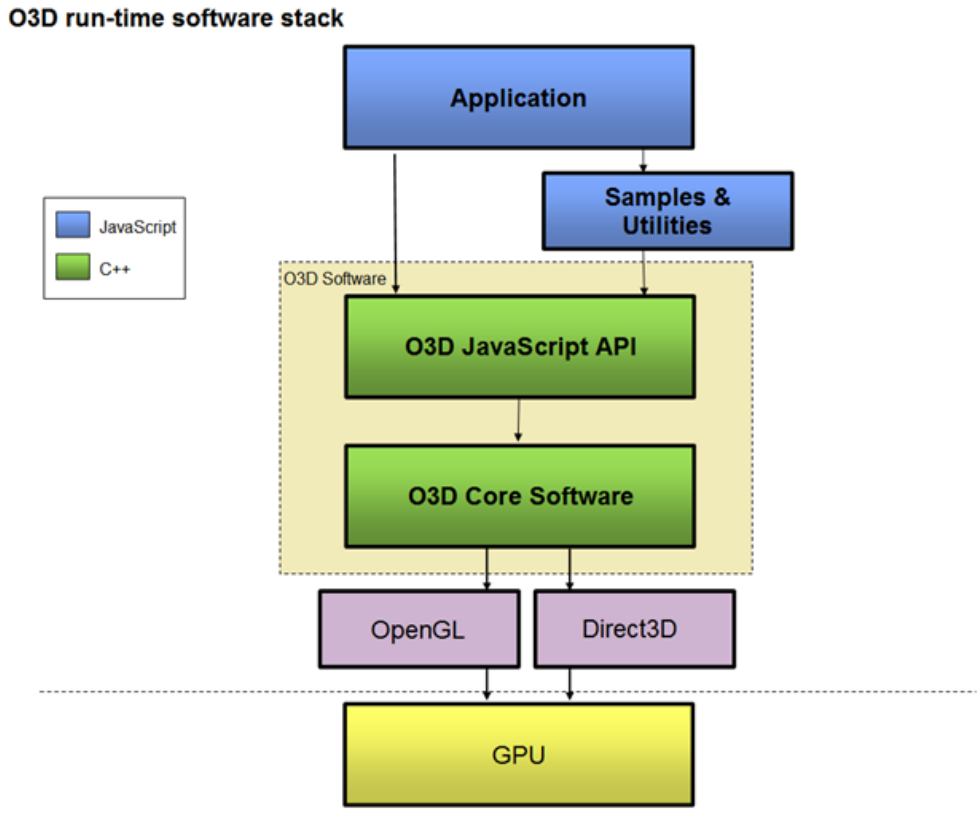

Figura 1 - Arquitetura da API O3D.

A API O3D possui um *plugin*, que deve ser instalado no computador local. Esse *plugin* faz a comunicação com as bibliotecas *OpenGL* e *Direct3D*. O desenvolvedor utiliza *javascript* como linguagem de programação para fazer uso da API. Esta API é bastante poderosa, pois disponibiliza para uso todos os recursos complexos do *OpenGL* e do *Direct3D*, como por exemplo, efeitos de reflexos e sombras (Joamp 2010).

2. Motivação para Desenvolvimento do *Framework*

Esta API trabalha de forma assíncrona com a página aberta pelo navegador, um conceito semelhante ao AJAX (W3schools 2010). Por ser recente, esta API ainda não possui um *framework*, ou *engine* próprios, o que torna o desenvolvimento de jogos para O3D um processo extenso e complexo (Google Code 2010).

Este trabalho propõe iniciar um projeto de *framework* que englobe a maior parte possível do processo de desenvolvimento de jogos e busca abstrair ao máximo a programação direta na API (Figura 2).

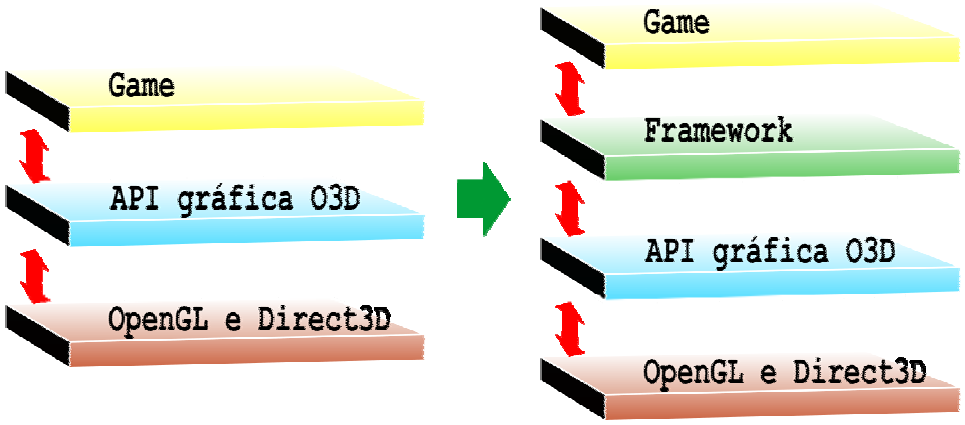

Figura 2 - Framework como uma camada de abstração.

3. Projeto e Desenvolvimento do *Framework*

Por se tratar do ambiente WEB, foi utilizado *javascript*, *PHP* e *MySQL* para o desenvolvimento do *framework* pois essas ferramentas apresentam enorme popularidade atualmente na área de internet, além de serem livres de licenças comerciais para utilização. (O'Reilly 2005). Contudo, em algumas áreas do *framework*, foram utilizadas classes de terceiros, com licença livre, para suprir necessidades específicas. Essas classes serão apresentadas nos tópicos a seguir.

3.1. Levantamento das necessidades

Depois de estudado o desenvolvimento de jogos utilizando a API, encontram-se necessidades inerentes ao processo de desenvolvimento de jogos que serão

citadas a seguir como os recursos que irão constituir o *framework*, sem enfocar aspectos mais avançados como a implementação de ambientes *multiplayer*, inteligência artificial e *engines* de física. Esses recursos são:

- **Controle de projetos –** consiste em armazenar informações sobre os projetos que são desenvolvidos e permitem que o desenvolvedor alterne entre os projetos de forma simplificada. As informações sobre os modelos, elementos, sons, texturas, cenário, personagens estão associados a um projeto criado pelo desenvolvedor e são armazenadas em banco de dados. Existe, ainda, a possibilidade de salvar e restaurar projetos em qualquer fase do desenvolvimento do jogo.
- **Interface gráfica do *framework* -** a elaboração de uma interface gráfica para o *framework* é uma tentativa de aproximar mais desenvolvedores e pessoas que ainda não possuem uma experiência na área de desenvolvimento para a realidade das possibilidades da nova tendência da internet. Na interface existe uma área de visualização em tempo real das alterações do projeto, como inclusão de elementos no cenário e posicionamento dos objetos.
- **Importação de elementos externos -** automatização do processo de importação de elementos 3D externos mediante poucos cliques.
- **Integração da API de áudio -** importante recurso para o desenvolvimento de jogos, ausente na API O3D, é o de reprodução de áudio. A *SoundManager* 2 (Soundmanager2, 2010), uma API de reprodução de áudio em *javascript* foi escolhida para fazer parte do *framework*.
- **Controle de comandos do jogo -** esta parte do *framework* visa a organizar, de forma estruturada, os vários eventos que podem acontecer durante o jogo. Um dos desafios do *framework* é automatizar o processo de associar os efeitos sonoros aos eventos visuais, por meio de uma interface simples. Permite ao desenvolvedor definir, por exemplo, qual animação o modelo 3D fará enquanto se mover e atribuir ao evento um efeito sonoro característico.

3.2. Modelagem

Foi desenvolvida uma modelagem para maior compreensão do funcionamento do *framework*. Os diagramas usados foram julgados necessários para melhor definição das funcionalidades do *framework*.

A modelagem do *framework* foi feita antes de seu desenvolvimento. Primeiramente, foi estudado o processo de desenvolvimento de jogos para maior compreensão das necessidades desse processo. Em seguida, foi feita uma análise da API O3D, e observou-se seus aspectos, sob a ótica do processo de desenvolvimento de jogos. O *framework* foi modelado a partir das necessidades levantadas após a análise da API. A Figura 3 apresenta o diagrama de contexto, com o funcionamento do *framework* num alto nível de abstração. Os elementos do jogo (imagens, áudio, modelos 3D etc.) são inseridos e armazenados, depois ministrados pelo desenvolvedor, que programa a lógica do jogo e obtém do *framework* a visão de um projeto estruturado.

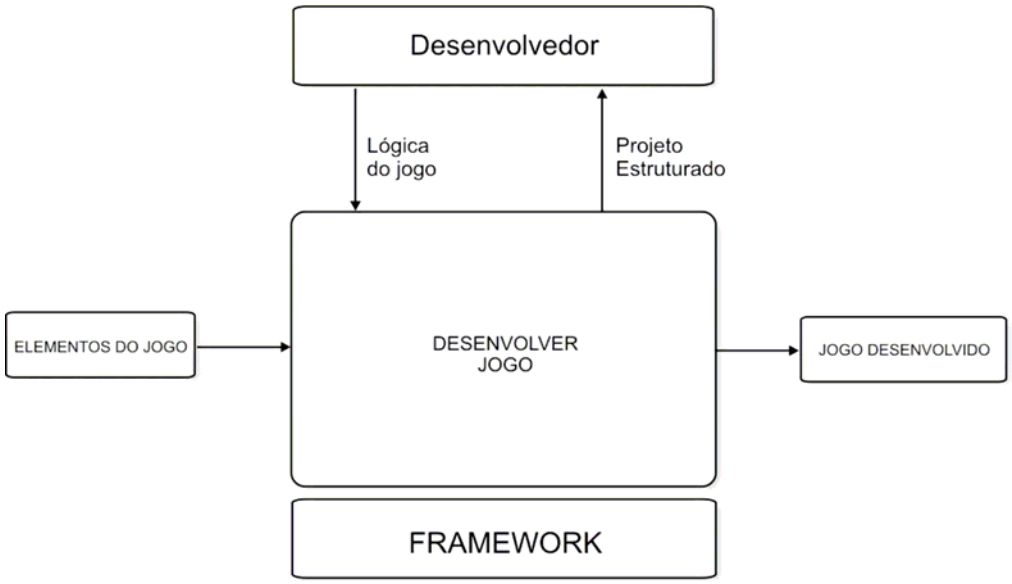

Figura 3 - Diagrama de Contexto.

O funcionamento básico do *framework* pode ser dividido nas seguintes etapas:

- 1.º O desenvolvedor (usuário) cria um novo projeto ou abre algum existente previamente salvo;
- 2.º Insere os elementos do jogo. Arquivos de som e imagem são armazenados diretamente na biblioteca do projeto. Já os elementos 3D devem sofrer um processo de conversão antes de serem armazenados;
- 3.º Com os elementos do jogo armazenados na base de dados, é permitido ao desenvolvedor configurar, da forma adequada, esses elementos, e criar os personagens, e associar comandos do teclado com movimentos do personagem. Pode, também, vincular os efeitos sonoros aos personagens, cenários e elementos 3D em geral; associar as texturas e definir o posicionamento das câmeras, do cenário e seus elementos;
- 4.º O desenvolvedor introduz o enredo do jogo e edita, manualmente, os *scripts*, programando os eventos e acontecimentos que irão ocorrer no jogo;
- 5.º A última etapa consiste em salvar todo o trabalho feito, testar e verificar se contém erros ou falhas que devem ser corrigidas. Com o projeto estruturado pelo *framework* e seus elementos armazenados em banco de dados, fica mais fácil a depuração e substituição de qualquer elemento do jogo.

3.3. Banco de Dados

Foi desenvolvido em *MySQL*, com a estrutura ilustrada na Figura 4. A tabela projeto é utilizada para identificar em qual projeto estão armazenados todos os elementos do jogo. O banco foi projetado para atender às necessidades do

cadastro de elementos de forma genérica. Os elementos do cenário são armazenados na tabela que pode possuir diferentes propriedades de acordo com tipo do elemento. Informações sobre os arquivos da biblioteca do projeto estão na tabela-arquivo.

O código gerado pelo *framework* é armazenado na tabela *script*. Esse código será utilizado para publicar o projeto para distribuição.

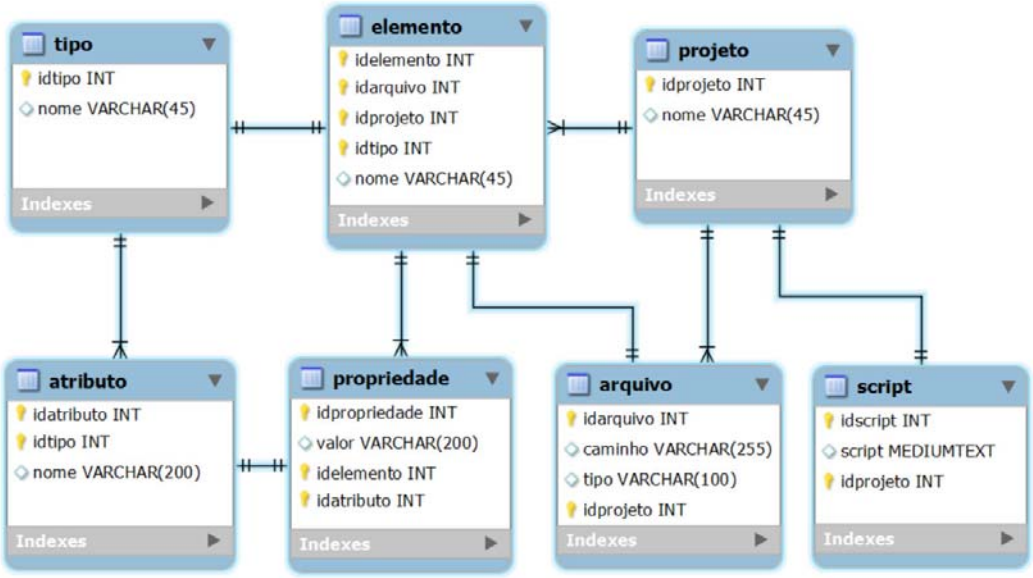

Figura 4 - Modelo do banco de dados.

3.4. Tela de Cadastro de Projetos

A primeira tela do *framework* é a de cadastro de projetos, ilustrada na Figura 5. Nesta tela, o usuário pode inserir, abrir, editar no nome de um projeto ou excluir um projeto. Essa é a única tela que é exibida fora do ambiente de trabalho.

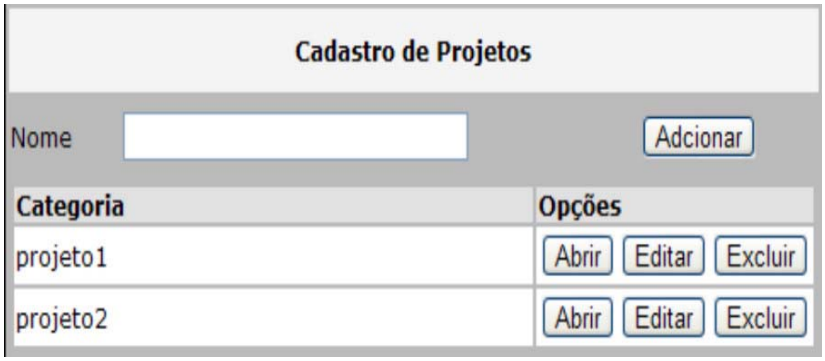

Figura 5 - Tela de cadastro de projetos.

3.5. Ambiente de Trabalho

A Figura 6 é a tela principal do *framework*, o ambiente de trabalho. Essa tela possui uma barra de menus, que dá acesso às demais telas.

Foi utilizado o *framework javascript prototype*, de Sam Stephenson, para exibir e manipular janelas sobrepostas no ambiente de trabalho. Esse recurso otimiza o desempenho do projeto, pois elimina a necessidade de recarregar o ambiente de trabalho, com seus gráficos, toda vez que houver alguma mudança de tela. Além disso, promove maior comodidade ao usuário, que pode manipular diferentes telas do sistema no mesmo ambiente de forma simultânea. Na barra lateral, o usuário tem diversas formas básicas para inserir na cena, além de possuir *links* diretos para a tela de manipulação de câmera e de luz. Além da barra de menus e da barra lateral, a tela é dividida em 4 quadros, sendo um, de propriedades, dois com imagens em 2D, chamados de quadros de posicionamento, e um, com gráfico em 3D.

O quadro de propriedades exibe os atributos do elemento selecionado. Estes podem ser posição, tamanho, textura etc. Os quadros de posicionamento representam as diferentes formas de visualização do cenário, sendo estas: de cima (eixos x e z) e de frente (eixos x e y). É possível alterar o posicionamento dos objetos nesses quadros, com utilização do *mouse*, ao clicar e arrastar o elemento para a posição desejada. Estes dois quadros estão sincronizados, de forma que, quando um elemento é movido em algum quadro sua posição é atualizada nos demais. O quadro em 3D representa o resultado em 3D das ações do usuário com uma área de visualização, somente. Para gerar e manipular os gráficos em 2D foi utilizada uma classe de licença livre, baseada no SVG do *javascript.* (Raphael 2010).

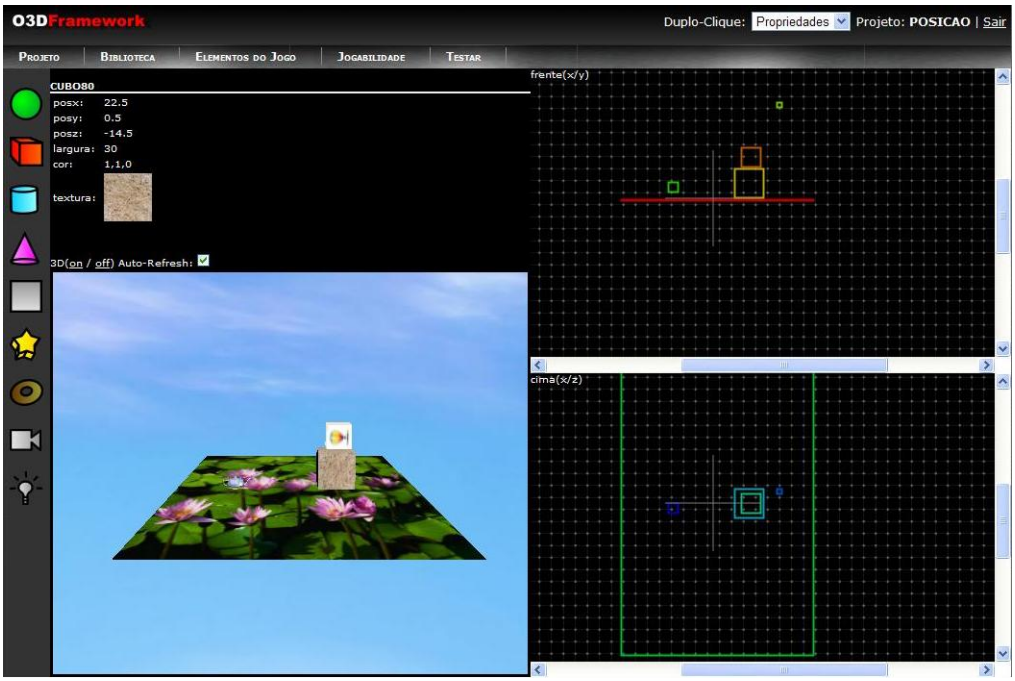

Figura 6 - Tela Ambiente de trabalho.

3.6. Tela Importar para a Biblioteca

Nessa tela é possível importar para a biblioteca do projeto, arquivos de som, imagens e modelos 3D. Som e imagens são simplesmente armazenados, já os modelos 3D devem sofrer um processo de conversão, caso não estejam no padrão de modelos da O3D. Os arquivos importados são listados abaixo do componente de *upload*, em que é possível excluí-los, se necessário. A Figura 7 mostra a tela de importação.

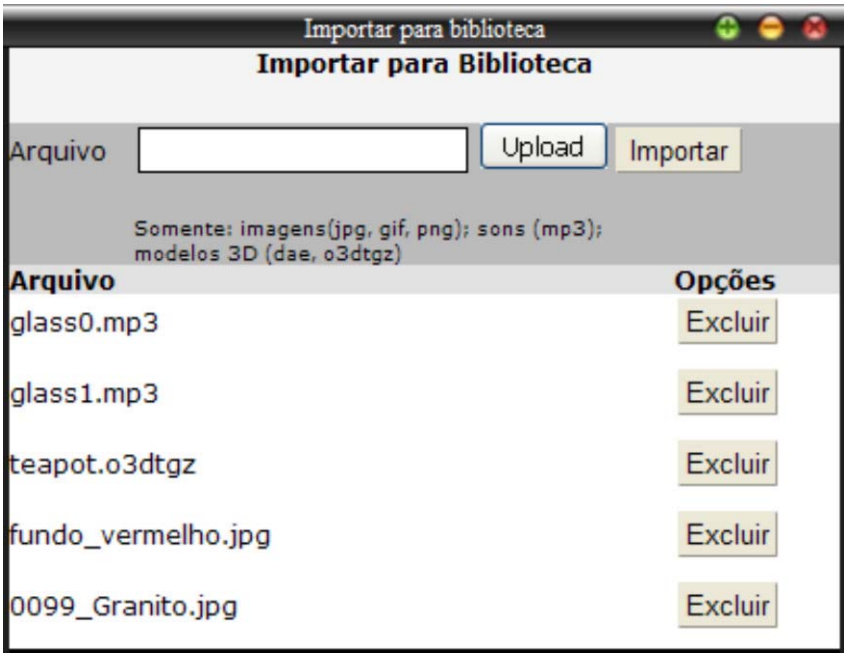

Figura 7 - Tela Importar para a biblioteca.

3.7. Tela Entidades

Nela o usuário poderá incluir no cenário diferentes tipos de entidades, como modelos 3D importados, elementos 3D primitivos, pontos de iluminação e o próprio cenário. Na figura 8 ilustra-se essa característica.

Na área “Criar Elemento” da tela de entidades, o campo NOME refere-se ao nome da entidade que será criada, o campo TIPO determina qual será o tipo da entidade, que pode ser, por exemplo, cubo, esfera, cilindro etc. Caso o tipo de elemento escolhido seja “modelo”, o campo ARQUIVO será habilitado para que o usuário selecione o modelo 3D, correspondente importado na biblioteca. Logo a seguir, estão listadas as entidades criadas e suas propriedades. Nessa área, é possível visualizar o valor de cada propriedade de uma entidade, modificar esse valor e salvar as alterações.

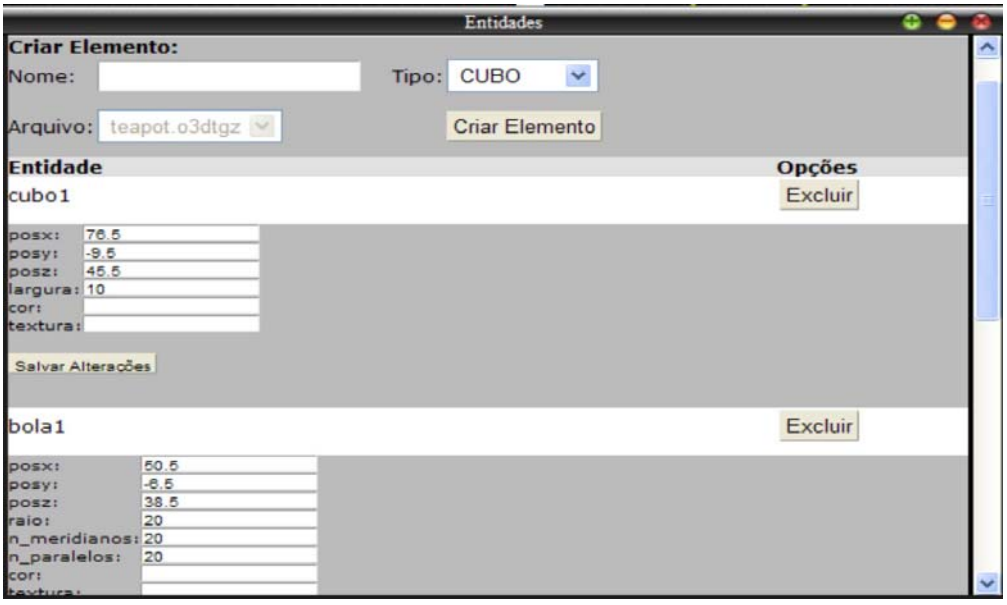

Figura 8 - Tela de Entidades.

3.8. Tela Cenário

Consiste em uma grande esfera com a textura voltada para dentro, e que cria o aspecto de céu. A esfera tem origem no ponto [0,0,0] do ambiente 3D. Na tela Cenário, tem-se o campo profundidade, que especifica o raio da esfera, e o campo textura, cuja imagem irá compor o interior da esfera. O cenário deve ser grande o suficiente para englobar todos os elementos. A Figura 9 mostra a tela do cenário, em que é possível alterar as propriedades do cenário.

Figura 9 - Tela do Cenário.

3.9. Tela Câmera

Seu objetivo é facilitar a troca de câmera, a qualquer momento, durante o desenvolvimento do jogo, e utilizar as câmeras pré-definidas do *framework*. Os modos de câmera pré-definidos são: FPS, terceira pessoa e 2D. A Figura 10 mostra a tela de câmera.

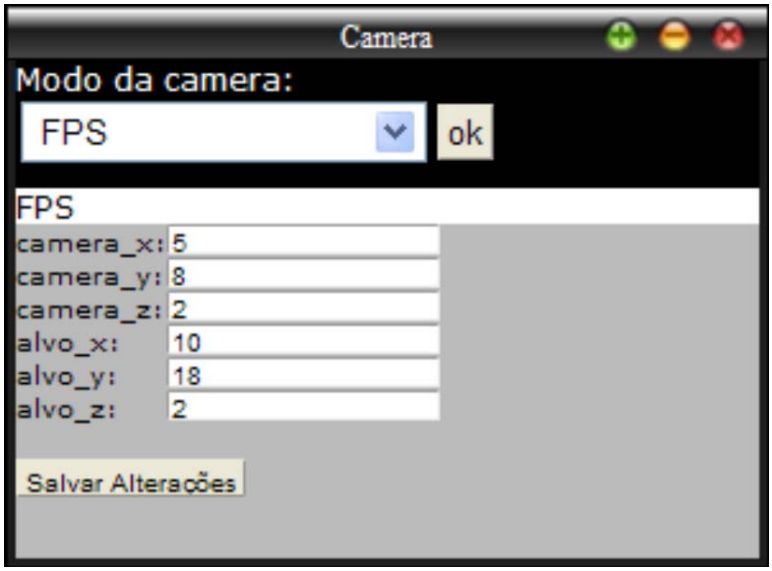

Figura 10 - Tela de Câmera.

3.10. Tela Editar *Script*

Todo código gerado pelo *framework* é armazenado e pode ser editado na tela editar *script*. O usuário poderá modificar o código, de acordo com suas necessidades. A Figura 11 ilustra a tela de editar *script*.

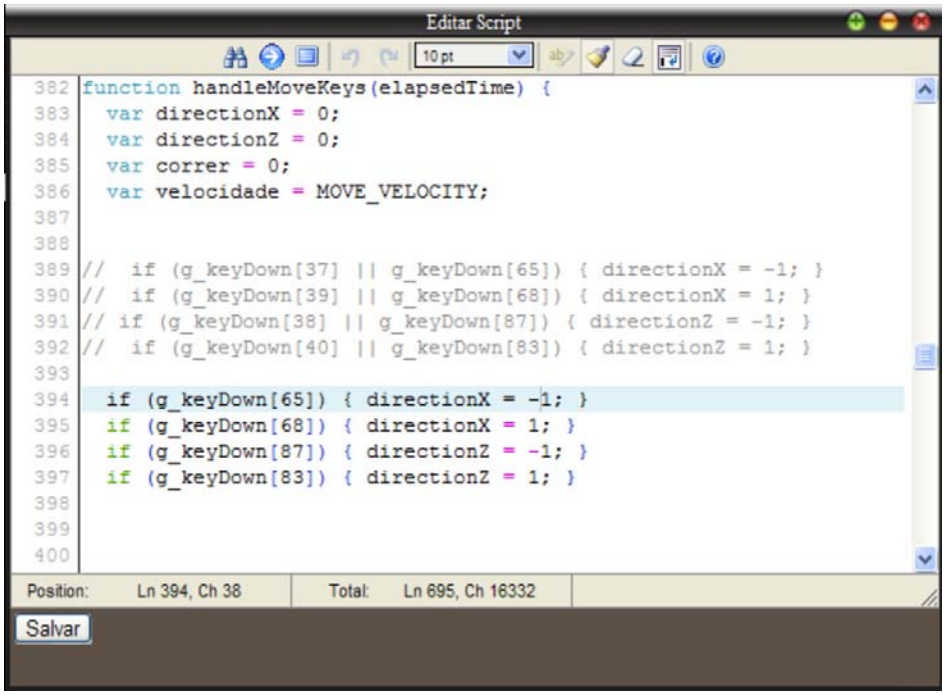

Figura 11 - Editar Script.

A fim de melhorar a experiência do usuário nesta tela, e de permitir recursos de sintaxe colorida, buscar palavra-chave no código, aumentar a fonte, entre outros

recursos, utilizou-se uma classe de edição de código em *javascript* com licença livre, o *EditArea*, de Christophe Dolivet.

3.11. Tela Renderizar

A tela renderizar cria o ambiente 3D, com o código gerado pelo *framework*, em uma janela dentro do ambiente de trabalho. A Figura 12 mostra a tela renderizar aberta no ambiente de trabalho.

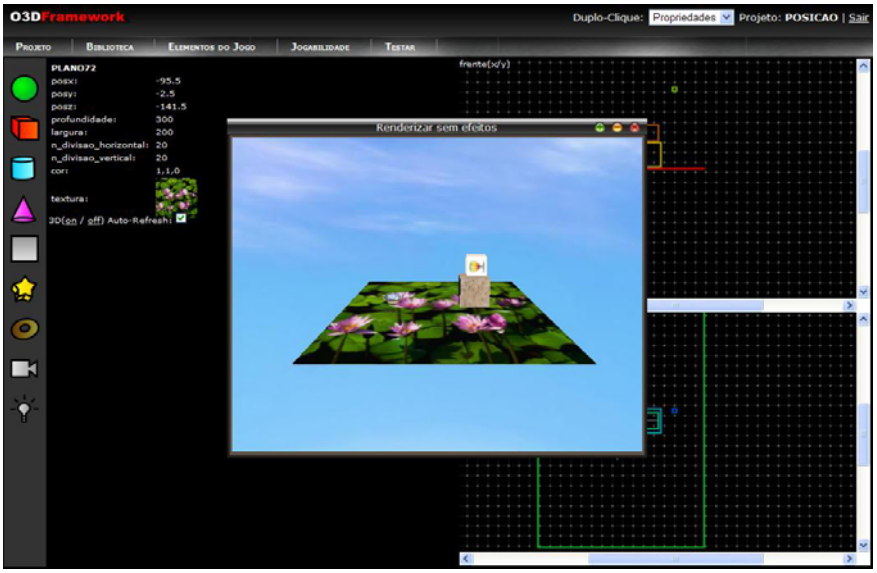

Figura 12 - Tela renderizar aberta no ambiente de trabalho.

3.12. Tela Animação

É responsável por cadastrar os intervalos de *frames* dos modelos 3D, que contém algum tipo de animação. Essas animações poderão ser utilizadas posteriormente, na Tela de Comandos. A ideia é reproduzir alguma animação durante um comando de ação. A Figura 13 mostra a Tela de Animação.

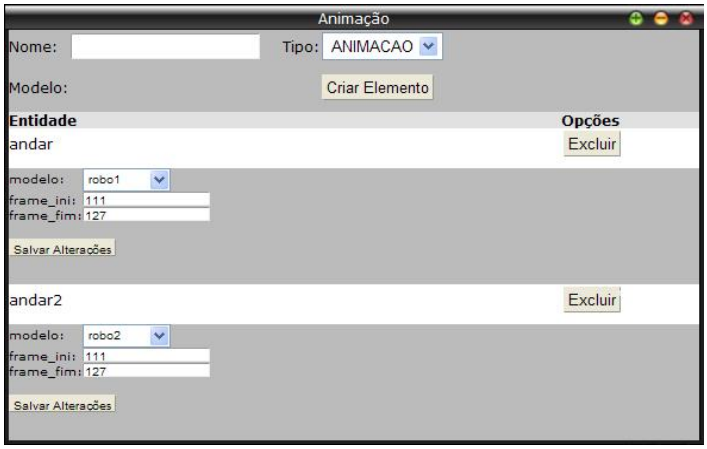

Figura 13 - Tela Animação.

3.13. Tela Comandos

É onde os comandos do jogo são configurados de maneira genérica. O desenvolvedor pode criar um novo comando, e atribuir-lhe o nome. No momento, o único tipo de comando implementado é o "Movimento", que permite deslocar os elementos pelo cenário. Por exemplo, pode ser criado o comando "andar" do tipo "movimento". Após sua criação, é necessário configurar propriedades e definir o modelo que sofrerá a ação, a animação configurada anteriormente, na tela de Animação, (se existir), a velocidade de deslocamento, o áudio que será reproduzido durante a ação (som de passos, por exemplo) e atribuir o código ASCII das teclas às variáveis de deslocamento nos eixos x, y e z. Então, ainda no exemplo, caso a tecla "W" seja associada à variável de deslocamento "x+", o elemento sofrerá incremento na sua posição referente ao eixo x durante o jogo, se a tecla "W" for pressionada. Da mesma forma na tecla "S" para na variável "x-", quando esta for pressionada durante o jogo, o elemento sofrerá decremento na posição referente ao eixo x. A Figura 14 mostra a tela de comandos.

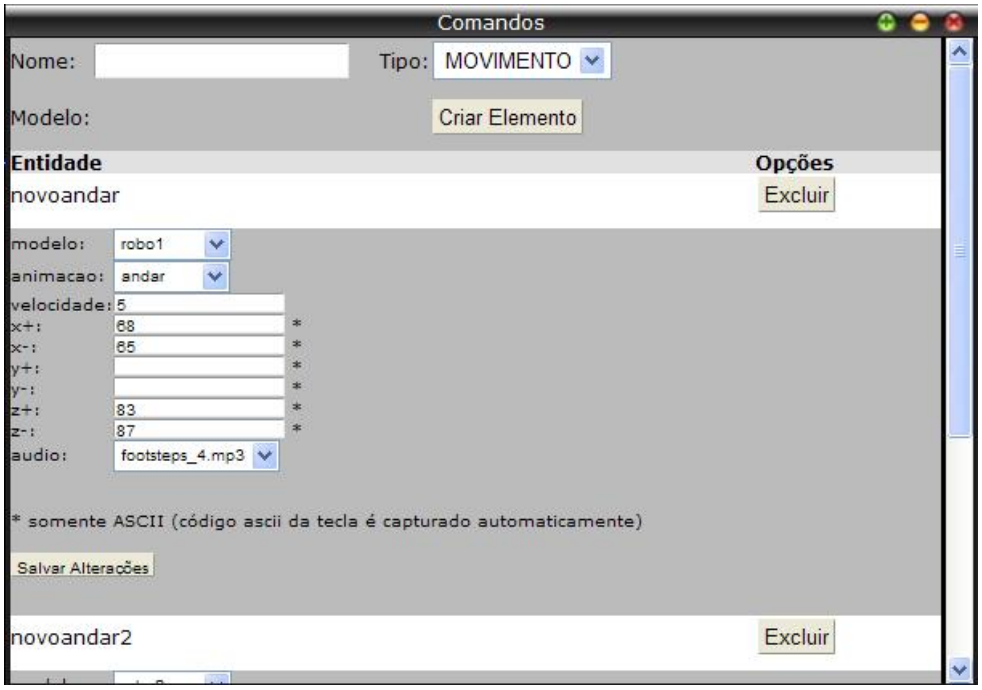

Figura 14 - Tela de Comandos.

4. Conclusão

Ao fazer um comparativo entre a API e *framework* para construir um protótipo de jogo, com cenário e personagem, o uso do *framework* reduz bastante o tempo de trabalho. Se levarmos em conta o processo de armazenar os elementos do jogo, posicionar esses elementos corretamente no cenário, atribuir movimento e animação aos personagens, concluímos que o *framework* atende ao proposto, abstraindo a API. Para realizar esses procedimentos o desenvolvedor não necessita sequer ver o código gerado. Outros recursos ainda podem ser adicionados ao *framework*, como um módulo de física que manipule colisões, gravidade, velocidade, peso e outros atributos físicos essenciais para o desenvolvimento de jogos modernos.

Referências

Clua, Esteban Walter Gonzalez, Bittencourt, João Ricardo (2005). Desenvolvimento de Jogos 3D: Concepção, Design e Programação.

Google Code (2010). API O3D. Disponível em http://code.google.com/intl/pt-BR/apis/o3d/. Último acesso em 01 de Outubro de 2010.

Joamp (2010). Jogl, Java Binding for the OpenGL API. Disponível em http://jogamp.org/jogl/www/. Último acesso em 01 de Outubro de 2010.

O'reilly, Tim (2005). O que é web 2.0? Padrões de design e modelos de negócios para a nova geração de software.

Raphael (2010). Biblioteca Javascript. Disponível em http://raphaeljs.com/reference.html. Último acesso em 01 de Outubro de 2010.

Santos, Rangel Jungles, Battaiola, Andre Luiz, Dubiela, Rafael Pereira (2004). Aspectos Fundamentais da Criação de Jogos em Shockwave 3D.

Soundmanager2 (2010). Sound Manager 2 Documentation. Disponível em http://www.schillmania.com/projects/soundmanager2/doc/. Último acesso em 02 de Outubro de 2010.

Uol (2009). Teorias e Conceitos sobre a internet 3D. Disponível em http://codigofonte.uol.com.br/artigo/diversos/teorias-e-conceitos-sobre-a-internet-3d. Último acesso em 04 de Outubro de 2010.

W3schools (2010). Ajax Tutorial. Disponível em http://www.w3schools.com/ajax/default.asp. Último acesso em maio de 2010.