Utilizando o 3D Studio Max como Level Editor para Construção de Cenários para Ogre3D

Jorge L. Salvi Maikon C. Santos Daniel M. Tortelli Jacques D. Brancher

Universidade Regional Integrada do Alto Uruguai e das Missões – Campus de Erechim, Brasil

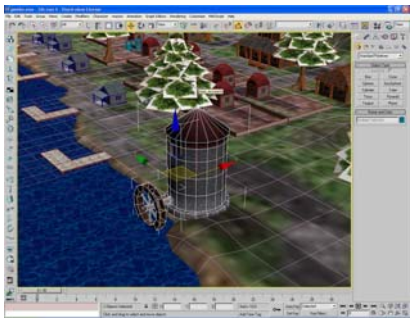

Figura 1: Exemplo de uso do 3D Studio Max como *level editor*.

Abstract

The aim of this paper is to present the strategy adopted for the construction of dynamic and static sceneries for the game "Taltun: A Terra do Conhecimento". The proposal was to develop sceneries using the modeling software (figure 1), the 3D Studio Max, and export them directly to the game engine without the necessity to develop or use a special level editor to elaborate the game levels.

Resumo

O objetivo deste artigo é apresentar a estratégia adotada para construção de cenários dinâmicos e estáticos para o jogo "Taltun: A terra do conhecimento". A proposta foi elaborar cenários através do uso do programa de modelagem (figura 1), o 3D Studio Max, e exporta-los diretamente para a *engine* do jogo sem a necessidade de se desenvolver ou fazer uso de um editor de fases especial para elaborar as fases do jogo.

Keywords: editor de fases, jogos de computador, física em jogos.

Authors' contact:
jlsalvi@hotmail.com
{maikoncs, daniel.beka}@gmail.com
jacques@uri.com.br

1. Introdução

A utilização de editores de fases (*level editor*) é o meio mais comum para o desenvolvimento de fases em jogos de computador. Segundo Rouse III [2000] o editor de fases tem grande importância no desenvolvimento de um jogo porque permite ao *designer* montar suas fases através da movimentação dos objetos em 2D enquanto estes podem ser visualizados em um mundo 3D, obtendo assim, uma perspectiva do final da fase do jogo.

Com a utilização dos editores de fases é possível inserir elementos gráficos (estáticos ou dinâmicos), luzes, câmeras entre outros elementos que podem compor a cena. Ainda, os editores possibilitam manipular os objetos da cena, para que eles se ajustem ao ambiente a ser criado. Segundo Clua e Bittencourt [2005] as manipulações básicas para edição dos objetos são as seguintes: seleção, translação, rotação e escalonamento.

Como geralmente um editor de fase é desenvolvido especialmente para uma única *engine* e seu desenvolvimento consome um tempo considerável, torna-se fundamental, em projetos de curto espaço de tempo, que os programadores tenham seu tempo enfocado na codificação da *engine*. No entanto, uma maneira de contornar este problema é adaptar as ferramentas de trabalho já existentes para a confecção dos cenários, neste caso a ferramenta de modelagem 3DS Max.

Nas próximas seções serão apresentados as ferramentas utilizadas para o desenvolvimento do jogo em estudo. Após, serão apresentados o *plugin* e o *loader* implementados para a adaptação da ferramenta 3DS Max ao propósito deste trabalho e, por último, as considerações finais.

2. Trabalhos Relacionados

A maioria das *engines* de jogos populares possuem editores de fases para possibilitar aos usuários a criação e/ou modificação das características dos ambiente que irão fazer parte do jogo. Segundo El-nasr e Smith [2006] processo de modificação e criação de fases é conhecido como *modding*.

Entre os editores de fases mais conhecidos podemos citar os dos jogos WarCraft III [2006] e Half Life 2 [2006] e da *engine* 3D Game Studio [2006]. Todos possuem características idênticas em seus

editores, mas, além disso, podem possuir também outros artefatos que implementam suas funcionalidades, tais como editores de *triggers* e de sons.

Já na *engine* Yake [2006], o uso do 3DS Max torna-se fundamental. Isto ocorre devido ao fato dela conter *plugins* responsáveis por exportar as propriedades do cenário modelado em um arquivo XML. A partir disso, o *loader* de gráfico (OGRE [2006]) e de física (ODE [2006] ou Novodex [2006]) buscam as propriedades exportadas a fim de construir a cena de forma idêntica àquela modelada.

Neste mesmo ramo também podemos ressaltar o trabalho realizado por Watsa [2001] no qual é explicado como o 3DS Max é utilizado para a construção de fases para a *engine* de seu jogo. Para isso foram usadas as características existentes no 3D Max e customizadas através da criação de *plugins* e scripts, os quais tornam possível a criação de fases para jogos de computador.

3. Ferramentas Utilizadas

Para a produção do jogo Taltun [2006], desenvolvido pelo projeto RPGEDU [2006] (*Role Playing Game* Educacional), são envolvidas diversas ferramentas. No entanto, restringe-se neste trabalho a apresentação das ferramentas relacionadas à construção de cenários.

Na construção gráfica dos cenários é utilizado o programa 3D Studio Max [2006] ou 3DS Max. Este é um programa de modelagem 3D que permite criar animações e renderizar imagens, sendo usado em produção de filmes de animação, vinhetas, comerciais para TV, maquetes eletrônicas e na criação de jogos eletrônicos.

Para a renderização dos gráficos gerados pela ferramenta citada acima dentro da *engine* do jogo é usado o OGRE (*Object-Oriented Graphics Rendering Engine*). Segundo Maior [2005], "O propósito do OGRE não é ser um *game engine*; ele é um *rendering engine* genérico que pode ser incorporado a outras bibliotecas..." dando assim suporte ao desenvolvimento de jogos 3D.

Ainda, para dar maior dinamismo e realismo ao jogo, é utilizada uma API de física: a OPAL [2006] (*Open Physics Abstraction Layer*). Sua principal característica é manipular fenômenos físicos envolvendo articulação de corpos, gravidade, detecção de colisão, atrito e outros fenômenos físicos envolvendo objetos em mundos virtuais.

A OPAL proporciona uma interface de alto nível para motores de física de baixo nível utilizados em jogos e em outras aplicações. Atualmente na versão 0.4 suporta somente a biblioteca ODE (*Open Dynamics Engine*) e em atual desenvolvimento dará suporte a TrueAxis [2006] (*True Axis Physics SDK*).

4. O 3DS Max como Editor de Fases

Para Watsa [2001] uma das razões para se utilizar o 3DS Max como editor de fases é a reutilização das suas janelas de visualização. Outro motivo de seu uso é o aproveitamento das propriedades dos objetos modelados e o uso dos eventos de manipulação.

Entretanto, para se produzir um cenário completo para o jogo são utilizados os recursos citados no parágrafo anterior. Também é necessário seguir uma seqüência de passos especificados conforme a Figura 2.

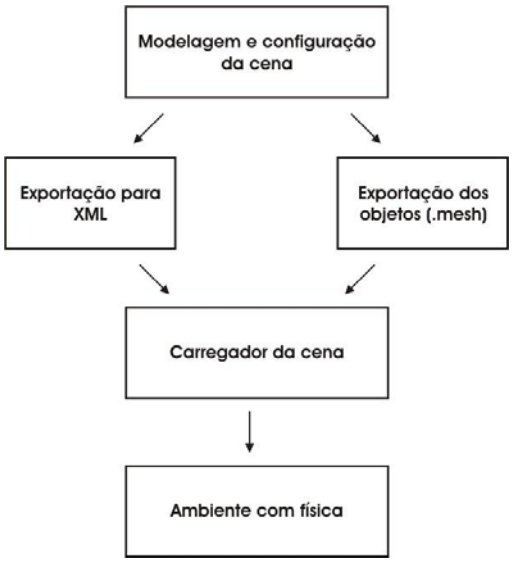

Figura 2: Fluxo para criação de um cenário.

O processo é iniciado com a modelagem, texturização e configuração da presente cena que em seguida será exportada através dos *plugins* do 3DS Max. Com isso serão gerados os arquivos que contêm as informações dos objetos e da cena a fim de serem carregados pelo *loader* da *engine*.

Com o final do processo de *loader*, obtém-se um ambiente com objetos dinâmicos e estáticos e com as propriedades e coordenadas especificadas no 3DS Max.

A seguir será exibido o método adotado para criação de cenários. Este está dividido em duas partes: o *plugin* e o *loader* da *engine*.

4.1 O plugin

Para que os objetos modelados usados na construção de cenários contenham as propriedades necessárias para atribuir física e gráficos foi preciso desenvolver um *plugin* para o 3DS Max. Este *plugin* foi desenvolvido para funcionar com o 3D Studio Max 6 e está dividido em duas partes: o modificador e o exportador, conforme apresentado na Figura 3. Cada

um deles trabalha de forma distinta, sendo que o primeiro é responsável por adicionar novos atributos ao objeto, já o último exporta os atributos dos objetos para posterior utilização pela *engine*.

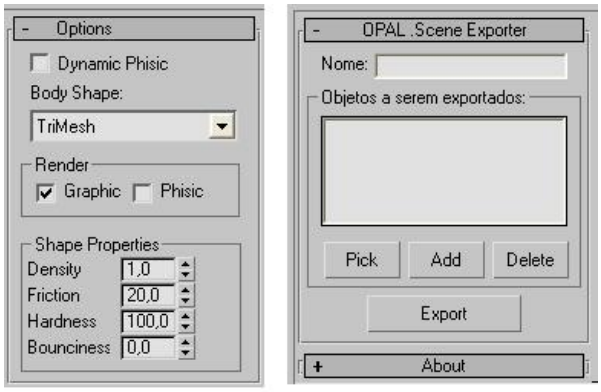

Figura 3: Modificador do 3DS Max para adição das propriedades da física (à esquerda) e o exportador (à direita).

Quando a modelagem dos objetos estiver concluída é necessário adicionar à sua lista de modificadores o elemento "Opal Scene", que se refere à primeira parte do *plugin*. Com este modificador são acrescentados novos atributos ao objeto, que servem para configurar as propriedades da física a ser aplicada ao objeto quando carregado ao jogo.

Um exemplo da utilização das propriedades da física dispostas pelo *plugin* é criação dos limites por onde o personagem pode andar pelo ambiente. Estes limites são feitos através de caixas de colisão, que formam uma espécie de labirinto, no qual o jogador apenas poderá navegar entre elas, conforme apresentado na figura a seguir.

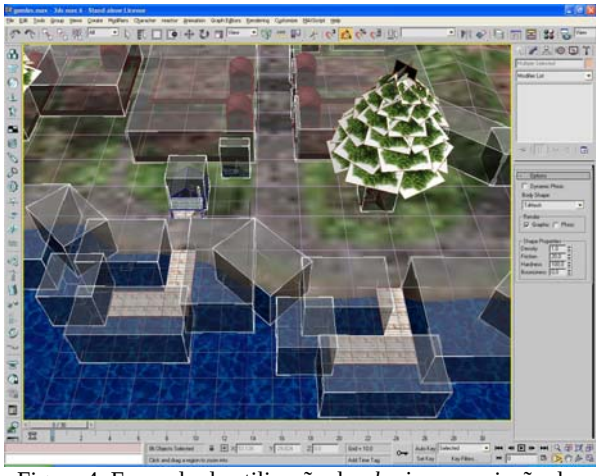

Figura 4: Exemplo de utilização do *plugin* para criação dos limites para navegação do personagem.

Após a configuração de todos os objetos que irão compor a cena então é feito o uso da segunda parte do *plugin*: o exportador. Este tem como escopo gerar um arquivo XML que conterá as informações aplicadas com o modificador e também informações relacionadas coordenadas gráfica do objeto em questão. Um exemplo de XML gerado pelo *plugin* pode ser visto na Figura 5.

```
<scene>
    <nodes>
        <node name="CavPilar" id="0" >
            <position x="-161.323" y="10.258" z="-177.267" />
            <rotation qx="0.0" qy="0.0" qz="0.0" qw="1.0" />
            <scale x="0.375095" y="0.412079" z="0.412079" />
            <dimensions x="914.655" y="49.8475" z="858.08" />
            <radius r="0" />
            <entity
                    name="CavPilar"
                    meshFile="CavPilar.mesh"
            />
            <physic
                    geometric="none"
                    dynamic="false"
                    type_simulation="graphic_no_physic"
                    hardness="0"
                    friction="0"
                    bounciness="0"
                    density="0"
            />
        </node>

        <node name="next . . ." id="1" >
        </node>

    </nodes>
</scene>
```

Figura 5: Exemplo de XML gerado pelo *plugin* de exportação.

Conforme observado acima, cada objeto será representado por um nodo dentro da cena. Este nodo contém atributos referentes à sua posição, rotação, escala, dimensão e raio (quando necessário). A partir destes atributos é que o objeto especificado na *tag* entidade irá se dispor no mundo virtual.

A *tag* de física especifica que tipo de geometria (caixa, esfera ou malha) será utilizada para a colisão dos objetos. Ainda é definido se o objeto será dinâmico ou estático, o tipo de renderização (somente gráfico, somente física ou ambos) e outras propriedades referentes ao comportamento do objeto durante a simulação.

Contudo, ainda falta exportar a parte que compõe a geometria do objeto. Para isto é utilizado o *plugin* "3D Studio Max Exporter for meshes and animation" da Ogre, gerando também um arquivo XML que em seguida é convertido para a extensão ".mesh" pelo programa OgreXMLConverter.

4.2 O loader

Logo após a exportação das especificações para a criação do ambiente virtual, contidos no arquivo XML gerados pelo *plugin*, começa a etapa final do processo, no qual carrega as informações contidas no arquivo e cria os objetos que irão compor a cena conforme o que foi modelado no 3DS Max.

Para a criação dos nodos (objetos) do ambiente a ser montado, o *loader* busca a informações dos objetos contidas no arquivo XML, fazendo a leitura dos dados e armazenando em memória as informações de cada nodo para que os mesmos possam ser criados no futuro.

A criação dos nodos pode ser feita de modo flexível e dinâmico, tendo a possibilidade de criar, tanto objetos em tempo de execução da aplicação (durante o jogo), quanto no carregamento da cena. Além disso, o *loader* fornece suporte para criação de todos os nodos em um único comando ou ainda através do índice ou nome de cada elemento.

O processo de criação de um nodo envolve dois elementos imprescindíveis: o uso da OGRE, para os objetos gráficos, e da OPAL, para criação de sólidos utilizados na simulação da física. As informações contidas no XML é que determinarão como cada nodo vai ser criado.

A primeira fase de criação de um nodo é a criação do sólido que simula a física do objeto, atribuindo sua posição, orientação e escala. A partir deste ponto, então é definida a geometria do sólido e acrescentadas as propriedades da físicas (atrito, densidade, etc).

Ao final da etapa de criação do nodo é construída a entidade gráfica, correspondente à rederização do objeto, no qual é anexado o arquivo *.mesh* correlato ao nodo. Após gerado, a cena passa a conter mais um objeto, sendo ele estático ou dinâmico. Se dinâmico há a possibilidade do objeto se mover através da aplicação de uma determinada força. Assim sendo, a entidade gráfica deve ter as mesmas coordenadas do sólido que simula a física.

Por este motivo, o *loader* necessita ser atualizado dentro do *rendering* da aplicação, para que a física e a parte gráfica estejam coerentes, gerando a simulação gráfica dos objetos dinâmicos de uma forma sincronizada.

Uma outra característica que incorpora este *loader* é o gerenciamento dos objetos após sua criação, possibilitando atribuir e obter dados sobre os nodos, tais como posição, velocidade atual, entre outras propriedades gráficas e de física.

Após o processo de *loader*, o cenário modelado no 3DS Max (Figura 4) fica como ilustrado a seguir. Aqui as caixas de colisão não são renderizadas, porém as propriedades da física aplicada a elas estão ativas. Sendo assim, quando o personagem, representado por uma esfera dinâmica, entrar em intersecção com as caixas, então o personagem automaticamente fica bloqueado para andar naquela direção.

Figura 6: Exemplo de cenário após o *loader*.

5. Conclusão

O uso do 3DS Max como editor de fases juntamente com o *loader* desenvolvido proporcionou diversos benefícios para o desenvolvimento do jogo. Dentre eles podemos citar o melhor aproveitamento do tempo e de esforço de trabalho dos programadores, a criação de *triggers* de modo visual e o carregamento automático de objetos estáticos e dinâmicos para o jogo.

Apesar das vantagens apresentadas, o uso desta técnica impossibilita ao usuário criar e/ou alterar os cenários, devido a um maior grau de complexidade exigido para manusear a ferramenta e seus *plugins*.

Contudo, apenas o fato de efetuar a carga automática dos objetos estáticos e dinâmicos já torna este recurso útil, pois elimina a necessidade de se configurar manualmente as propriedades de cada objeto que irá compor o ambiente.

Como trabalhos futuros, pretende-se adicionar ao *plugin* a possibilidade de se criar câmeras e luzes. Outro ponto almejado é desenvolver articulações (*joints*) para os objetos que contenham física, fazendo com que haja relacionamento forçado entre dois corpos, como é o exemplo de uma dobradiça.

Agradecimentos

Agradecemos ao apoio prestado pela FINEP e pela URI – Campus de Erechim que tornaram possível a realização deste trabalho. Parte integrante do projeto RPGEDU – FINEP 1925/2004-0.

Referências

3D GAME STUDIO. Disponível em: http://www.3dgamestudio.com/ [Acessado em 18 Ago. 2006].

3D STUDIO MAX. Disponível em: http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112 [Acessado em 13 Ago. 2006].

CLUA, E. W. G. e BITTENCOURT, J. R. 2005. Desenvolvimento de Jogos 3D: Concepção, Design e Programação. *In: XXIV Jornadas de Atualização em Informática (JAI) Part of XXIV Congresso da Sociedade Brasileira de Computação*, 22-29 jul. 2005 São Leopoldo. São Leopoldo: UNISINOS, 1313-1357.

EL-NASR, M. S. e SMITH, B. K. 2006. Learning Through Game Modding. *ACM Computers in Entertainment*, Vol. 4, No. 1, Jan. 2006. Article 3B.

HALF LIFE 2. Disponível em: http://www.half-life2.com/ [Acessado em 17 Ago. 2006].

MAIOR, T. S. et al. 2005. O Engine Gráfico OGRE 3D. *In: SBGames 2005* parte do *IV Workshop Brasileiro de Jogos e Entretenimento Digital* , 23-25 Nov. São Paulo. CD-ROM.

NOVODEX Disponível em:http://www.novodex.com [Acessado em 1 Ago. 2006].

ODE. Disponível em: http://www.ode.org [Acessado em 1 Ago. 2006].

OGRE. Disponível em: http://www.ogre3d.org/ [Acessado em 12 Ago. 2006].

OPAL. Disponível em: http://ox.slug.louisville.edu/~o0lozi01/opal_wiki/index.php/Main_Page [Acessado em 18 Ago. 2006].

ROUSE III, R. 2000. Toiling with Tools. *In: 27th International Conference on Computer Graphics and Interactive Techniques*, 25-27 Jun. Lousiana - USA. New York: ACM Press, 5-9.

RPGEDU. Disponível em: http://www.uri.com.br/rpgedu/ [Acessado em 18 Ago. 2006].

TALTUN: A terra do conhecimento. Disponível em: http://www.malisoft.com.br/taltun/webstandards/ [Acessado em 20 Ago. 2006].

TRUEAXIS. Disponível em: http://trueaxis.com [Acessado em 18 Ago. 2006].

WARCRAFT III. Disponível em: http://www.blizzard.com/war3/ [Acessado em 18 Ago. 2006].

WATSA, S. 2001. *Case Study: Using Max Script for Building Game levels* [online] Gamasutra - The Art & Business of Making Games. Disponível em: http://www.gamasutra.com/features/20010824/watsa_01.htm [Acessado em 18 Ago. 2006].

YAKE - VR AND GAME ENGINE. Disponível em: http://www.yake.org/ [Acessado em 15 Ago. 2006].