

Construindo Cenários de Jogos Móveis Utilizando o 3DS Max e a API M3G

Bruno M. Arôxa Artur F. Mittelbach

C.E.S.A.R., Brasil

Abstract

This paper presents a 3D scene building process for mobile games. The level design built in 3D Studio Max tool is exported to a properties file which is loaded and mounted during the game – that relies on the M3G (Mobile 3D Graphics) API. Memory constraints (inherent from mobile device) were addressed providing a repository of 3D objects used to dynamically compose the game scenes. In a similar way, the “rendering window” technique was applied in order to improve game performance.

Resumo

Este artigo apresenta um processo para a construção de cenários de jogos 3D para dispositivos móveis. O *level design*, feito no 3D Studio Max, é exportado em um arquivo de propriedades que é carregado e montado sob demanda durante a execução do jogo (que utiliza a API M3G – *Mobile 3D Graphics*). As limitações de memória (características neste tipo de dispositivo) foram contornadas com a definição de um repositório de modelos utilizado para a composição dinâmica dos cenários. De maneira semelhante, a técnica de “janela de renderização” foi aplicada para otimizar o desempenho do jogo.

Keywords: construção de cenários, level design, dispositivos móveis, 3ds max, m3g

Authors' contact:

{bruno.aroxa, arturfm}@gmail.com

1. Introdução

Segundo Phill Co [2006], o *level design* é uma parte integral do desenvolvimento de um jogo. O projetista do nível precisa trabalhar junto ao time de arte para criar experiências as mais interessantes possíveis para o jogador. Além disso, ele precisa estar em contato com o *game designer* para garantir o balanceamento ideal da dificuldade do jogo, proporcionando um melhor sentimento de progresso e realização para o jogador.

Na busca por soluções para resolver os problemas característicos de jogos para dispositivos móveis (restrições de memória e baixo poder de processamento), e automatizar o processo de carga dos cenários no jogo, foram estabelecidas algumas etapas bem definidas durante o desenvolvimento do jogo. No contexto deste trabalho, esse conjunto de etapas está

sendo citado como “processo”. Como estudo de caso, o processo proposto foi aplicado na construção dos cenários (e *level design*) de um jogo 3D de naves, rodando em um dispositivo sobre a plataforma Java (CLDC [2008] 1.1, MIDP [2008] 2.0 com suporte à API M3G [2008]).

Este artigo está estruturado da seguinte forma: a seção 2 cita outros trabalhos relacionados com o tema proposto; a seção 3 apresenta as ferramentas envolvidas na construção dos cenários; a seção 4 detalha as etapas do processo de construção dos cenários; finalmente, a seção 5 expõe as conclusões e considerações finais deste trabalho.

2. Trabalhos Relacionados

O projeto de motores para exportar cenários do 3DS Max para jogos 3D não é uma idéia nova. Watsa [2001] propôs a utilização do Max Script para gerar níveis para jogos dentro do próprio 3DS Max. Para o contexto do estudo de caso feito por Watsa, a proposta foi de modificar o 3DS Max, através de *plugins* e *scripts*, para automatizar tarefas de criação de níveis.

Outro trabalho mais recente [Salvi et al. 2006] propõe a utilização do 3DS Max como editor de níveis para jogos 3D utilizando o motor de processamento gráfico OGRE [2008]. O trabalho utilizou como estudo de caso o jogo para PC “Taltun: A terra do conhecimento”.

Apesar da proposta de utilizar o 3DS Max como editor de nível ser similar aos trabalhos citados, este trabalho se difere tanto no aspecto de exportação dos níveis como no processo de carga dos cenários, que precisa satisfazer exigências da API 3D para dispositivos móveis.

3. Ferramentas Utilizadas

O desenvolvimento de jogos requer uma gama variada de ferramentas que vão desde ferramentas de controle de versão (CVS – *Concurrent Version System*, SVN – *Subversion*, etc.) até ferramentas de modelagem e edição de imagem. Esta seção abordará apenas as ferramentas envolvidas na construção de cenários e no *level design*.

3.1. 3DS Max

O 3DS Max [2008] foi escolhido como ferramenta para criação dos níveis por cinco motivos básicos:

- O projeto já possuía licença do software;
- O time já tinha know-how da ferramenta;
- O time já tinha know-how dos exportadores M3G [2008];
- Vasta biblioteca de material de pesquisa;
- O suporte acessível à criação de scripts de exportação.

3.2. Max Script

O Max Script é a ferramenta de *scripts* própria do 3DS Max. Através dela é possível criar, ler e redefinir atributos para os objetos da cena, bem como exportá-los para arquivos de propriedades. Além disso, a ferramenta também permite a criação de interfaces gráficas para facilitar a execução dessas tarefas.

O Max Script foi escolhido pela sua vasta biblioteca de referência [Max Script 2008], integração total com o 3DS Max e flexibilidade.

3.3. M3G Exporter

Para o processo de exportação do repositório de peças, feito com base no formato M3G, foi utilizado o pacote de ferramentas de desenvolvimento disponibilizado pela Mascot Capsule [2008]. O pacote inclui:

- *Micro3D Plugin*: para exportação através do 3DS Max;
- *M3G Converter*: Ferramenta para a conversão do arquivo exportado pelo *Micro3D Plugin* (.h3t) para o formato carregado pela API M3G no jogo;
- *M3G Viewer*: Ferramenta de visualização de arquivos no formato m3g.

Apesar de algumas limitações, a ferramenta foi escolhida por falta de uma ferramenta alternativa que se aproximasse das necessidades do projeto

4. Processo para construção dos cenários

O processo de criação de cenários adotado neste trabalho apresenta seis etapas bem definidas que são ilustradas na Figura 1.

Na primeira etapa são criadas as peças que servirão de base para o *level design*. Com o repositório estabelecido, segue-se a segunda etapa que diz respeito à construção do cenário propriamente dito. As etapas seguintes envolvem as exportações dos cenários e do repositório, preparando os artefatos necessários para a carga do cenário no jogo. Ao final do processo, o *level design* é validado com o intuito de proporcionar a melhor experiência possível para o usuário.

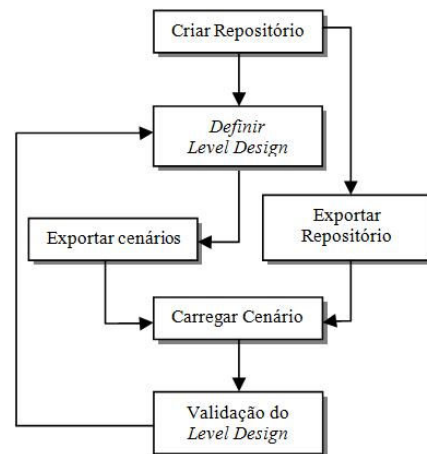


Figura 1: Etapas do processo de criação de cenários

4.1. Criar Repositório

Uma das preocupações nessa etapa é de projetar peças que possam ser usadas em situações diferentes, sem que se perceba a repetição, fato que é possível com as manipulações básicas – translação, rotação e escalonamento [Clua e Bittencourt 2005] – disponíveis no 3DS Max.

Para construção de estruturas inorgânicas como os prédios do jogo foram usados sólidos básicos, tais como: cubos, paralelepípedos, etc. Para a criação de figuras orgânicas de cenário como montanhas, arbustos, corais, etc., foi modelada uma peça irregular simples, porém, que permitisse a sua utilização em diferentes contextos. Para facilitar futuras referências, essa peça metamórfica, cuja estrutura básica é apresentada na Figura 2, será chamada de *brush*.

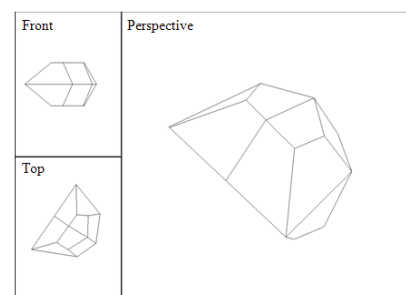


Figura 2: Sólido geométrico, base para composição dos cenários

4.2. Definir Level Design

O processo de concepção do *level design*, no contexto deste jogo, seguiu o seguinte fluxo:

1. *Definir o tema do nível*: A definição do tema dá suporte para a arte conceitual do cenário;
2. *Definir o skyline do nível*: Diz respeito às características do cenário de acordo com o tema definido previamente;
3. *Montar a estrutura básica do cenário*: Definir a sequência de terrenos, utilizando as peças do

repositório, com base na duração estimada do nível;

4. *Montar elementos orgânicos*: Definir os elementos estáticos do cenário utilizando modificações do *brush*;
5. *Inserir marcadores para inimigos*: Definir cada inimigo bem como seus atributos (velocidade, inteligência, formação, etc.);
6. *Inserir marcadores para itens coletáveis*: Posicionar os itens coletáveis de acordo com a geografia e dificuldade do nível.

4.3. Exportar Cenário

O processo de exportação do cenário consiste unicamente na extração de propriedades relevantes dos objetos no cenário, de forma a possibilitar a sua reconstrução no ambiente do jogo. Para diminuir a complexidade do problema, o cenário foi dividido em unidades lógicas, chamadas *terrenos*. Os terrenos, por sua vez podiam conter ou não objetos 3D. Para cada objeto foram atribuídas as seguintes propriedades:

- Indexador que o associa ao terreno;
- Identificador do tipo de objeto;
- Coordenadas de posição;
- Valores de rotação;
- Informação de escala;
- Identificador de textura.

Além destas propriedades, padrão para todos os objetos, os objetos dinâmicos (inimigos, itens coletáveis, etc.) acumulavam ainda um conjunto de atributos específicos (velocidade, tipo, etc.).

A extração das informações dos objetos para um arquivo de propriedades foi feita utilizando o Max Script – ferramenta de *scripts* que integra o próprio 3DS Max., o respeitando um protocolo simples, especificado pela própria equipe para definir a ordem de escrita das propriedades. A definição desse protocolo foi fundamental para garantir a leitura correta dos arquivos de propriedades na etapa de carga do cenário, durante o jogo. A Figura 3 apresenta um exemplo de nível exportado.

```
23,05,1.0,0.5,1.0,133.445,0.0,-1.0,0.0,-30.5798,54.576,11607.1,1
23,05,1.0,1.37294,1.0,60.8267,0.0,-1.0,0.0,-120.228,-20.6941,11620.6,1
23,05,1.0,1.97294,1.0,55.8509,0.0,1.0,0.0,70.6496,-6.15476,11677.3,1
23,05,1.78331,1.0,1.0,59.7451,0.0,-1.0,0.0,-96.4743,1.58681,11733.5,-1
23,05,1.78331,1.0,1.0,59.7451,0.0,-1.0,0.0,-85.3158,1.58681,11740.0,-1
23,800,1.0,1.0,1.0,0.0,0.0,0.0,0.0,15.4688,44.2274,11783.3,0
```

Figura 3: Entradas de um arquivo de nível exportado

A entrada em destaque diz que será adicionado no terreno 23 (primeira propriedade da linha) que um *brush*, cujo tipo está indicado pela segunda propriedade, com as características de escala, rotação e posição especificadas nas propriedades seguintes. A última propriedade define a textura que será aplicada ao *brush*.

4.4. Exportar Repositório

O repositório exportado (arquivo *.m3g*) contém um conjunto de elementos que podem ser carregados dinamicamente no jogo através da API 3D. Para a visualização desses arquivos foi utilizada a ferramenta M3G Viewer, que faz parte do pacote de ferramentas disponibilizado pela Mascot. A Figura 4 apresenta o esquema de visualização da estrutura da árvore de nodos gerada pelo exportador.

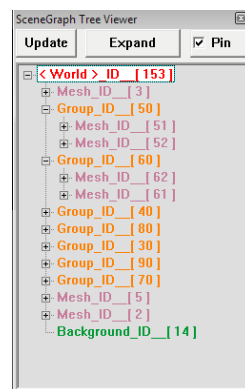


Figura 4: Interface gráfica do exportador

Um dos maiores benefícios trazidos por este visualizador é a possibilidade da validação dos objetos antes mesmo da carga do arquivo na aplicação. Desta forma, os problemas no processo de exportação eram identificados com antecedência, tornando o trabalho mais produtivo.

4.5. Carregar o Cenário

Durante a inicialização de cada fase do jogo, o cenário é carregado com base nas propriedades dos terrenos, exportadas nas etapas anteriores. Para otimizar o desempenho do jogo e o uso de memória, a carga dos objetos é feita utilizando uma janela de renderização, que garante um limite máximo de objetos 3D carregados na cena. A Figura 5 apresenta um diagrama esquemático da janela de renderização, que para o contexto do jogo, com os devidos ajustes de câmera (*near clip*, *far clip*, *FOV – Field of View*, etc.), apresentou resultados satisfatórios com apenas três terrenos.

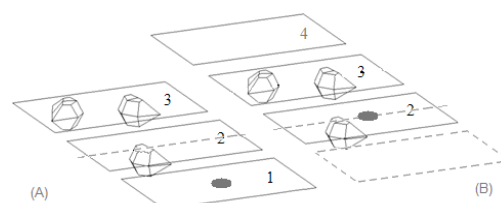


Figura 5: Diagrama esquemático da janela de renderização com três terrenos

A progressão do jogo sobre o cenário é medida pela projeção da câmera sobre o terreno (representado pelo ponto escuro na Figura 5 (A)). Quando a projeção alcança a metade do terreno seguinte (2), os objetos do terreno anterior (1) são liberados para o próximo

terreno não carregado (4) seja alocado, como é apresentado na Figura 5 (B).

Durante o processo de carga de cada terreno, o motor recupera a peça no repositório de acordo com o seu tipo e aplica as transformações necessárias, com base no arquivo de propriedades do nível. Para os objetos dinâmicos (inimigos, itens coletáveis, etc.) são analisados também propriedades específicas como: IA, velocidade, estratégia, etc. Como boa parte do tratamento dos dados é feita durante a exportação dos cenários, o módulo de carga do cenário no jogo acabou sendo bastante enxuto, com apenas duas classes para controle e entidades e um *parser* para fazer a tradução dos níveis exportados.

4.6. Validar Level Design

A validação do *level design* é feita levando-se em consideração alguns dos aspectos de jogabilidade definidos por Phill Co [2006]:

- Desafios justos (jogos muito fáceis ou muito difíceis acabam por frustrar o jogador);
- Sistema de recompensa que motive o usuário a querer jogar;
- Objetivos dentro do alcance do jogador.

Apesar de saber que idealmente um bom *level design* deve apresentar as características citadas, na prática o resultado da avaliação acaba sofrendo influência de fatores externos como: TTM – *Time to Market* e limitações dos aparelhos (viabilidade técnica).

5. Conclusão

A utilização do processo para a construção de cenários, proposto neste artigo, se mostrou bastante satisfatória e eficaz durante o desenvolvimento do jogo. Os resultados obtidos apresentaram um ganho expressivo no tamanho final do jogo. O repositório exportado consumiu em torno de 30kb e aproximadamente 15kb (valor médio) por arquivo de propriedade (*level design*). Em testes realizados com a exportação do nível completo, um único cenário ficou com aproximadamente 115kb, fato que inviabilizou esta abordagem (visto que foram construídos nove cenários).

Além do ganho no tamanho do jogo, o uso do motor proporcionou um melhor uso dos recursos de memória do aparelho. Com a abordagem de janela de renderização, foi possível ter um controle mais fino da quantidade de objetos 3D carregados na memória.

Apesar dos pontos positivos mencionados, alguns desafios precisam ser melhorados, tais como: melhorar a GUI do exportador, permitindo a inserção e edição das propriedades dos objetos; reduzir a precisão dos valores das propriedades exportadas, que atualmente

estão com cinco casas decimais, para apenas duas (que já seriam mais que suficientes para o jogo desenvolvido), de forma a reduzir o tamanho dos arquivos de nível.

Agradecimentos

Os autores agradecem ao C.E.S.A.R [2008] pela excelente infra-estrutura provida para a execução e sucesso deste projeto.

Referências

- 3DS MAX, 2008. Disponível em: <http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112> [Acessado em 14 Ago. 2008].
- MAX SCRIPT, 2008. Disponível em: http://images.autodesk.com/adsk/files/maxscript_8_help.zip [Acessado em 14 Ago. 2008].
- C.E.S.A.R, 2008. Disponível em: <http://www.cesar.org.br> [Acessado em 12 Ago. 2008].
- CLDC – CONNECTED LIMITED DEVICE CONFIGURATION API, 2008. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html> [Acessado em: 03 Out. 2008].
- CLUA, E. E BITTENCOURT, J. R., 2005. Desenvolvimento de Jogos 3D: Concepção, Design e Programação. In: *XXIV Jornada 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.
- Co, P., 2006. *Level Design for Games: creating compelling game experiences*. Berkley, CA: New Riders, 2006.
- M3G API, 2008. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr184/index.html> [Acessado em: 14 Ago. 2008].
- MASCOT CAPSULE M3G TOOLKIT, 2008. Disponível em: <http://www.mascotcapsule.com/toolkit/m3g/en/index.php> [Acessado em: 14 Ago. 2008].
- MIDP – MOBILE INFORMATION DEVICE PROFILE API, 2008. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html> [Acessado em: 03 Out. 2008].
- OGRE, 2008. Disponível em: <http://www.ogre3d.org/> [Acessado em 18 Ago. 2006].
- SALVI, J. L., SANTOS, M. C., TORTELLI, D. M. E BRANCHER, J. D. 2006. Utilizando o 3D Studio Max como Level Editor para Construção de Cenários para Ogre3D. In: *V Workshop Brasileiro de Jogos e Entretenimento Digital parte do SBGames 2006*, 8-10 Nov. Recife.
- 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_pfv.htm [Acessado em 18 Ago. 2008].