

Criando níveis 3D para jogos móveis usando um editor 2D

Bruno M. Arôxa Luiz A. B. da Silva Artur F. Mittelbach

C.E.S.A.R, Brasil

Abstract

The main objective of this paper is to present a technique to ease the static building of 3D scenes for mobile games using the tile map editor, Mappy, as the level editor tool. The goals are to simplify the level edition using a 2D editing tool and to enable exporting directly to the game's engine. This approach cut out the need of a 3D level editor. Moreover, the technique enables the creation of low cost levels (in this case study 8KB per level and 27KB to the 3D repository).

Resumo

O objetivo deste artigo é apresentar uma técnica para facilitar a construção de cenários 3D estáticos em jogos móveis, utilizando Mappy Tile Map Editor – Mappy – como ferramenta de edição de níveis. A proposta é simplificar a edição de níveis utilizando uma ferramenta de edição 2D e exportá-lo diretamente para o motor do jogo. Essa abordagem elimina a necessidade de um editor de fases 3D. Além disso, a técnica permite a construção de níveis a um baixo custo (no estudo de caso deste trabalho 8kb por nível e 27kb do repositório de peças 3D).

Keywords: editor de níveis, jogos para dispositivos móveis, *level design*.

Authors' contact:

{bruno.aroxa, arturfm}@gmail.com
arturbotelho_4@yahoo.com.br

1. Introdução

A utilização de editores de fases (*level editor*) é o meio mais apropriado para o desenvolvimento de fases em jogos de computador. Isso porque permite ao *designer* montar as suas fases, visualizando-as em um mundo próximo ao que será visto no jogo, assim como permite a edição ágil de qualquer mudança que possa vir a ser feita.

Entretanto, dependendo do tipo do jogo abordado, podemos usar uma abstração e representar níveis 3D através de uma representação 2D. Apesar de o projetista ter que abrir mão da visualização completa do nível, ele ganha uma forma de edição na qual poderá capturar mais facilmente as informações necessárias para o motor do jogo recriar o cenário durante a sua execução.

O objeto de pesquisa deste documento ajuda a resolver alguns problemas referentes à facilidade e

rapidez na hora da criação e edição dos níveis do jogo. O Mappy [2008] oferece um ambiente com visão 2D simples e de fácil entendimento, que não requer conhecimento prévio de ferramentas 3D.

A técnica proposta aqui é baseada na geração de arquivos binários de nível onde estão presentes as informações necessárias para que o motor crie o mundo usando peças 3D e texturas presentes num repositório criado para o jogo. O uso dos arquivos citados proporciona uma excelente economia de espaço de memória do dispositivo. Este trabalho utilizará como estudo de caso um jogo 3D de corrida de carros, rodando em um dispositivo sobre a plataforma Java (CLDC [2008] 1.1, MIDP [2008] 2.0 com suporte à API M3G [2008]). Apesar de o estudo de caso ser realizado com um jogo de ambiente aberto, a técnica aqui proposta também pode ser aplicada à jogos de ambientes fechados que possam ser projetados em um plano 2D.

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 níveis; a seção 4 detalha as etapas do processo de criação dos níveis; finalmente, a seção 5 expõe as conclusões e considerações finais deste trabalho.

2. Trabalhos Relacionados

Até o momento os autores não encontraram nenhuma referência para trabalhos relacionados ao tema proposto. No entanto, outros trabalhos relacionados a editores de níveis foram apresentados, entre eles destacam-se: a proposta de um *framework* para a criação de editores de níveis [Filho et al. 2004]; e um trabalho relacionado ao uso de editores de níveis para a construção de cenários para jogos 3D, que foi proposto por Salvi et al. [2006]. Este último se diferencia no fato de utilizar um editor 3D e não 2D, como propõe este trabalho.

3. Ferramentas Utilizadas

A seguir serão descritas de forma sucinta as ferramentas relacionadas com o processo de criação de cenários e de *level design*, utilizadas neste projeto.

3.1 3DS Max

O 3DS Max [2008] foi usado nesse projeto para criar o repositório de peças que seria utilizado para a montagem dos níveis. O 3DS Max foi escolhido como

ferramenta para criação dos níveis por quatro motivos básicos:

1. O projeto já possuía licença do software.
2. O time já tinha know-how da ferramenta.
3. O time já tinha know-how dos exportadores 3DS – M3G [2008].
4. Vasta biblioteca de material de pesquisa.

3.2 Mappy Tile Map Editor

Foram realizados testes com dois editores gratuitos: Mappy, e o Tile Studio [2008]. Apesar de se apresentar como um produto aparentemente mais elaborado e estável, o Tile Studio não contava com todas as funcionalidades necessárias para suprir as necessidades do jogo. O Mappy foi escolhido, entre outros fatores, por permitir a edição de níveis em multicamada.

3.3 Lua Script

Lua [2008] é uma linguagem de *script* interpretada utilizada para expandir aplicações de forma geral. Ela é usada como forma de linguagem extensível que une partes de um programa feitas em mais de uma linguagem.

O fato de Lua ser uma linguagem de *script* suportada pelo Mappy, e apresentar características como portabilidade e simplicidade, direcionou a sua adoção no projeto.

4. Processo de criação dos níveis

O processo de criação dos níveis no contexto deste projeto seguiu etapas bem definidas, como é apresentado na Figura 1.

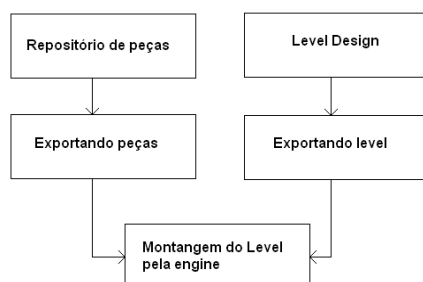


Figura 1: Etapas do processo de criação do nível do jogo.

As próximas subseções apresentam em detalhes cada fase do processo de criação e montagem dos níveis no jogo.

4.1 Definindo o Repositório de Peças

O artista 3D, com o apoio do projetista do nível, define os elementos necessários para compor o cenário do jogo. Por se tratar de um jogo de corrida ambientado num cenário urbano, vários dos elementos componentes de cenário são prédios e poderiam ser

tratados apenas com faces retangulares com aplicações de textura adequadas.

Grande parte do trabalho, entretanto, foi chegar ao conjunto mínimo de peças e formas de posicionamento para gerar o sentimento adequado de ambiente urbano. Durante esta fase foi necessário projetar e testar várias configurações de elementos gráficos e visualização para garantir a melhor configuração para o jogo. Entre elas:

1. *Ângulo das câmeras*: Garantindo que o jogador poderia ter visão ampla para o mundo do jogo e informação suficiente para tomar decisões relevantes para o jogo.
2. *Velocidade aparente*: Os primeiros testes com o jogo não apresentaram a sensação de velocidade esperada, apesar de os componentes da velocidade estar fisicamente coerentes. Para melhorar essa sensação foram feitos ajustes de câmera e aplicados efeitos gráficos tanto nos elementos da pista quanto nas texturas do mundo.
3. *Perspectivas e pontos de fuga*: Ajustes no posicionamento dos objetos da cena para proporcionar uma melhor visualização.
4. *Nível de detalhe das texturas*: Foram necessárias várias iterações para identificar o nível ideal de detalhe das texturas, de forma a obter o melhor custo-benefício para o jogo.

Após vários testes conseguiu-se reduzir o conjunto de peças do jogo para quatro peças:

1. Blocos de pista: peças que compõem as rodovias;
2. Limites (*guardrails*): peças que limitavam o movimento do jogador separando a área de jogo do cenário;
3. Cenário: composto basicamente por planos retangulares simples justapostos, com texturas diversas aplicadas: prédios, cercas gramadas, casas, lojas, delegacias, etc.;
4. Elementos especiais: Faixas de largada, placas de sinalização, pontos de som 3D, etc.

4.2 Exportando o Repositório de Peças

O repositório exportado (arquivo ‘.m3g’) contém o conjunto de peças utilizadas para a montagem dinâmica das pistas do jogo. Para a visualização desses arquivos foi utilizada a ferramenta M3G *Viewer*, que faz parte do pacote de ferramentas disponibilizado pela Mascot Capsule [2008].

Um dos maiores benefícios trazidos por este visualizador é a possibilidade da validação dos identificadores das geometrias antes mesmo da carga do arquivo na aplicação. Problemas no processo de exportação eram identificados com antecedência, o que tornou o trabalho mais produtivo.

4.3 Definindo o Level Design

O projetista do nível elabora os conceitos de cada um dos níveis já prevendo as devidas curvas de dificuldades e os sucessivos desafios do jogo. Com esses elementos em mãos, foram usadas diferentes camadas no Mappy para representar a sobreposição de alguns elementos, por exemplo, numa mesma área do mapa era possível representar uma pista, uma faixa de largada, uma placa de sinalização e uma parede de concreto como limitador. Foi necessário acostumar-se com a abstração e passar a usar símbolos comuns entre as camadas para representar elementos diferentes no jogo. A Figura 2 apresenta um exemplo de nível. No painel à direita, são apresentados os tipos de peças disponíveis para a construção.

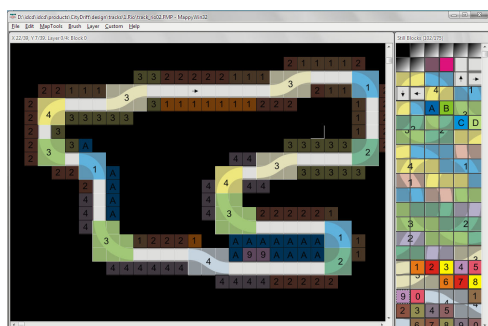


Figura 2: Edição de nível, tiles mapeados indicando cada peça e propriedade dos cenários.

Cada bloco no painel de peças possui um índice que o representa na matriz do nível. Uma vez definidos os parâmetros e necessidades básicas, esses índices foram mapeados manualmente para os tipos de peças modeladas no repositório.

Para este jogo, foram definidas as seguintes camadas:

1. *Camada de pista*: indicando qual peça do repositório seria utilizada para traçar a pista a ser corrida;
2. *Prédios (forma)*: usado para indicar a sucessão de planos de prédios fora da área de pista que simularia o ambiente urbano;
3. *Prédios (textura)*: uma vez definidos a posição e tamanhos dos planos/prédios foi possível referenciar a textura a ser aplicada neles;
4. *Camada de limites da pista*: indicando que tipo de separador seria usado entre a pista e o cenário: grades, canteiros de concreto, cercas vivas, etc.;
5. *Camada de som 3D*: indicando os pontos onde seriam aplicadas fontes de som 3D.

4.4 Exportando o Nível

Um *script* escrito em Lua foi utilizado em conjunto o Mappy, para gerar o arquivo binário contendo as informações necessárias para criar as pistas do jogo. Após finalizar a edição do nível, é realizado o processo de exportação onde é gerado um arquivo binário para a construção do cenário.

A representação visual do nível é interpretada pelo exportador como um conjunto de matrizes, onde cada camada do nível possui o conjunto de identificadores que serão utilizados pelo motor de montagem, conforme ilustra a Figura 3.

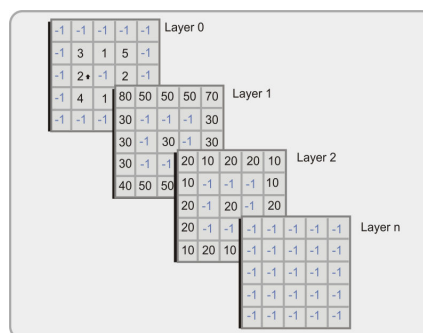


Figura 3: Estrutura de matrizes do nível.

Todas as camadas possuem as mesmas dimensões, e o identificador '-1' indica uma posição vazia na matriz. O arquivo binário está organizado da seguinte forma:

1. *Número total de camadas*: Representa o número total de camadas contidas no nível;
2. *Dimensão da matriz*: Define a largura e altura de todas as matrizes;
3. *Informações das matrizes*: Contém o conjunto de identificadores presentes na matriz da camada correspondente.

4.5 Carregando o Nível

A última etapa do processo de montagem do cenário consiste no mecanismo de carga do nível no motor de renderização 3D do jogo. Essa carga dos blocos que compõe cada pista é feita através do conjunto de classes apresentadas na Figura 4.

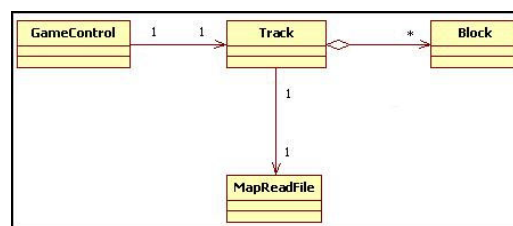


Figura 4: Representação UML das classes envolvidas no processo de carga do cenário no jogo.

A seguir são descritas as principais responsabilidades de cada classe:

- *GameControl*: Gerencia o fluxo de informação do jogo, delegando atividades para as devidas classes responsáveis.
- *Track*: Classe responsável por montar o nível. Inicialmente carrega as informações do nível através da classe *MapReadFile* e vai criando

novos blocos (instância de *Block*), que serão adicionados ao cenário do jogo.

- *Block*: Classe responsável por construir os blocos e elementos 3D que compõem o circuito do jogo.
- *MapReadFile*: Classe responsável pela leitura das informações contidas nos arquivos binários que são gerados pelo script.

O algoritmo de montagem faz uma busca na matriz, que é fornecida pela classe *MapReadFile*, para encontrar o identificador e os índices (x, y) do primeiro bloco a ser inserido no mundo 3D. O identificador encontrado é único e é o ponto de partida para inserir os blocos seguintes. Com o índice do primeiro bloco gravado, o algoritmo percorre a matriz em busca de novos identificadores, sempre analisando as posições vizinhas ao índice atual e, quando encontra um valor válido, cria o bloco que representa essa informação através da classe *Block* e insere no nível.

A inserção dos blocos no nível é feita com base no identificador da peça e na posição do bloco, de forma a garantir que cada bloco só será inserido no mundo uma única vez. A Figura 5 apresenta uma representação de como a peça é adicionada ao mundo.

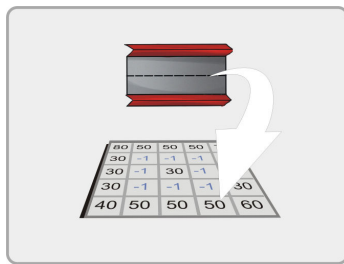


Figura 5: Representação de uma peça na matriz de nível.

5. Conclusão

A utilização do Mappy para abstrair a edição de nível de um jogo 3D para um ambiente 2D, reduziu a complexidade da criação e manutenção dos níveis, e se mostrou bastante eficaz para este trabalho.

Além disso, observou-se também uma redução expressiva do tamanho do jogo. Com a abordagem do repositório de blocos para a montagem dos níveis, foi possível oferecer uma grande quantidade (24 pistas) de níveis no jogo a um baixo custo (cerca de 8kb por nível e 27kb do repositório). Para o contexto do projeto, esse foi o maior benefício obtido neste trabalho.

Contudo, alguns problemas foram enfrentados durante a execução do projeto, expondo algumas limitações do modelo proposto. Entre essas limitações pode-se destacar o problema com o espelhamento de peças que, por uma limitação do motor gráfico, exigiu a duplicação de algumas peças; e o fato de só ser possível visualizar o resultado final do nível após a sua carga completa no jogo.

Outra limitação diz respeito ao alto acoplamento do motor de montagem ao domínio do jogo, requerendo um alto esforço para a adição de novas peças para compor os níveis, inviabilizando a sua utilização para outros domínios de jogos. Segundo Filho et al. [2004], esse alto acoplamento é advindo das características intrínsecas de cada jogo (características, regras, funcionalidades, etc.). Neste caso, uma melhoria clara para trabalhos futuros seria uma refatoração nas classes que compõem o módulo de carga, de forma a facilitar a inserção de novos tipos de peças.

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].
- C.E.S.A.R, 2008. Disponível em: <http://www.cesar.org.br> [Acessado em 12 Ago. 2008].
- CLDC – CONNECTED LIMITED DEVICE CONFIGURATION API. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html> [Acessado em: 03 Out. 2008].
- FILHO, W. M. A., RAMALHO, G. L. E FRANCA, A. C. 2004. GEF – Um Framework para Editores de Níveis. In: *III Workshop Brasileiro de Jogos e Entretenimento Digital* parte do *SBGames 2004*, 20-21 Out. Curitiba.
- LUA, 2008. Disponível em: <http://www.lua.org/> [Acessado em 25 Ago. 2008].
- M3G – MOBILE 3D GRAPHICS API. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr184/index.html> [Acessado em: 25 Ago. 2008].
- MAPPY EDITOR, 2008. Disponível em: <http://www.tilemap.co.uk/mappy.php> [Acessado em 25 Ago. 2008].
- MASCOT CAPSULE M3G TOOLKIT, 2008. Disponível em: <http://www.mascotcapsule.com/toolkit/m3g/en/index.php> [Acessado em: 25 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].
- TILE STUDIO, 2008. Disponível em: <http://tilestudio.sourceforge.net/> [Acessado em: 25 Ago. 2008].
- 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.