UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FERRAMENTA PARA GERAÇÃO DE MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D MMO

DISSERTAÇÃO DE MESTRADO

Fernando Bevilacqua

Santa Maria, RS, Brasil

2008

FERRAMENTA PARA GERAÇÃO DE MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D MMO

por

Fernando Bevilacqua

Dissertação apresentada ao Programa de Pós-Graduação em Informática
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Mestre em Informática

Orientador: Prof. Dr. Cesar Tadeu Pozzer (UFSM)

Dissertação de Mestrado Nº 2
Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

FERRAMENTA PARA GERAÇÃO DE MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D MMO

elaborada por
Fernando Bevilacqua

como requisito parcial para obtenção do grau de
Mestre em Informática

COMISSÃO EXAMINADORA:

(Presidente/Co-orientador)

Prof. Dr. Gerson Geraldo Homrich Cavalheiro (UFPel)

Profª Drª Iara Augustin (UFSM)

Santa Maria, 22 de Agosto de 2008.

“Não sabendo que era impossível, ele foi lá e fez.” — JEAN COCTEAU

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

FERRAMENTA PARA GERAÇÃO DE MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D MMO

Autor: Fernando Bevilacqua
Orientador: Prof. Dr. Cesar Tadeu Pozzer (UFSM)
Local e data da defesa: Santa Maria, 22 de Agosto de 2008.

Resumo em português aqui.

Palavras-chave: MMO, mundos virtuais, geração de terreno, jogos 3D.

ABSTRACT

Master's Dissertation
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

VXDL: A LANGUAGE FOR INTERCONNECTION AND RESOURCES SPECIFICATION IN VIRTUAL GRIDS

Author: Fernando Bevilacqua
Advisor: Prof. Dr. Cesar Tadeu Pozzer (UFSM)

Grid computing has been defined as an infrastructure integrator of distributed resources. Although it is already used on a large scale in many areas, this type of computational infrastructure is still an area of active research, with many open questions. Today, new research works investigate the application of resources virtualization techniques to perform the composition of virtual grids. These grids can be defined as a high level abstraction of resources (computing and network), through which users have a view of a wide range of interconnected computers, that can be selected and virtually organized. In a virtual grid, as well in a real grid, users and middleware must have tools that allow the composition and management of the infrastructure. Among these tools, there are languages for resource description that allow the specification of components that will be used in the infrastructure. In a virtualized environment, the resources descriptions languages should offer attributes that interact with some peculiarities, such as the possibility of allocate multiple virtual resources (computing and network) on the same physical resource. In this context, this work presents VXDL, a language developed for the interconnections and resources description in virtual grids. The innovations proposed in VXDL allow the description, classification and parameter specification of all desirable components, including network topology and virtual routers. VXDL also allow the specification of a execution timeline, which can assist grid middleware in the tasks of resources sharing and scheduling. To evaluate the proposed language, this work presentes I) a comparative study between VXDL and other resources description languages and II) an analysis of results obtained with the benchmarks execution in virtual infrastructures composed using different VXDL descriptions.

Keywords: virtualization, virtual grids, virtual clusters, resources description language.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

1.2 Objetivos e Contribuição

1.3 Organização do Texto

2 REVISÃO DE LITERATURA

2.1 Mundos pseudo-infinitos

A geração de conteúdo procedimentais é um assunto antigo no campo da computação gráfica. A aplicação desse tipo de técnica na geração de um mundo virtual completo foi utilizada por [7], cujo objetivo era gerar uma cidade virtual que fosse visualmente interessante e composta por construções complexas, porém cada uma delas sendo criada a partir de elementos mais simples.

Na abordagem utilizada, os autores dividiram o mundo virtual numa grade composta por diversos quadrados, chamados células. As coordenadas de localização de cada célula, em conjunto com uma semente global, são utilizadas como entrada para uma função de hash [10]. O resultado dessa função é utilizado como semente para um pseudo gerador de números aleatórios e irá definir todas as características das construções que estão dentro da célula. Dessa forma, o conteúdo de uma célula é sempre o mesmo, independente de quanto o usuário caminhe pelo mundo virtual e faça a célula em questão entrar ou sair do seu campo de visão. A figura 2.1 ilustra a divisão do mundo virtual em células.

Cada uma das construções existentes é gerada pela mescla de polígonos simples escolhidos aleatoriamente. Utilizando um processo iterativo, partindo do topo até a base,

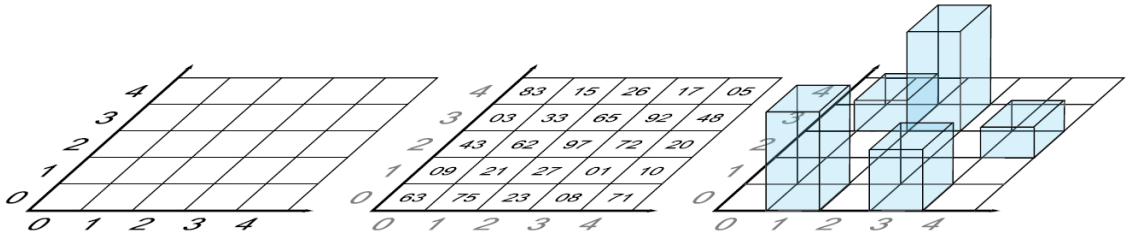

Figura 2.1: Mundo virtual dividido em células. Cada uma delas possui uma semente calculada com base em sua posição e com base uma semente global referente à cidade [7]

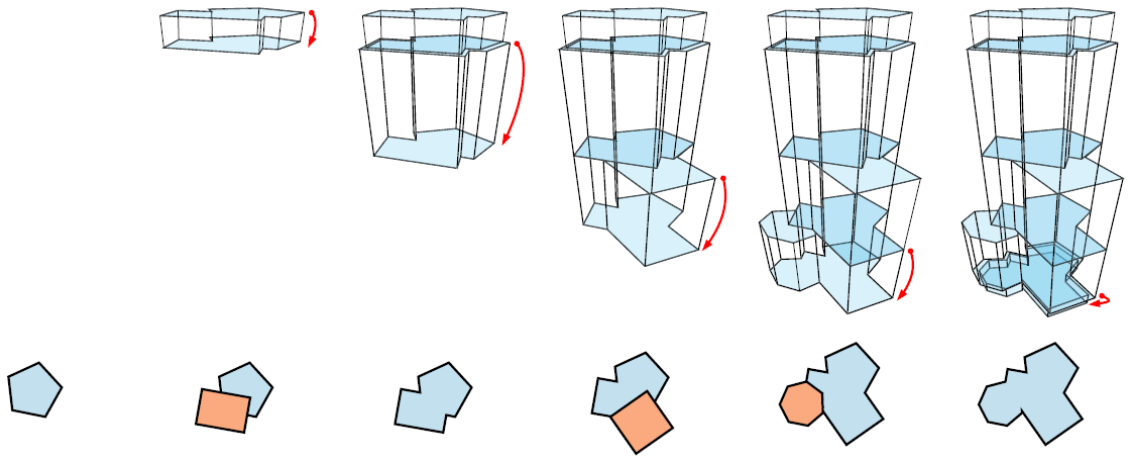

Figura 2.2: Geração de construções: em andar é gerado pela mescla de polígonos escolhidos e centralizados aleatoriamente [7]

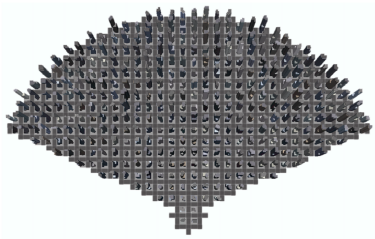

Figura 2.3: Campo de visão do usuário: apenas as células visíveis tem o seu conteúdo gerado [7]

em cada iteração o polígono escolhido é mesclado com o polígono anterior e, então, mais um nível (andar) é criado; esse processo garante que a construção, ao longo das iterações, cresça em altura e em largura de uma forma realística. Depois que a geometria da construção está pronta, ela é texturizada com janelas, sendo que o tipo da janela é escolhido aleatoriamente. A figura 2.2 ilustra a geração de construções.

Para garantir o uso racional de recursos computacionais, como memória e processamento, os autores utilizaram o conceito nomeado por eles de preenchimento por *view frustum*, que consiste em restringir à geração de conteúdo somente para as células que estão dentro do campo de visão do usuário. A medida que esse anda pela cidade virtual, novas células vão sendo adicionadas ao campo de visão e seu conteúdo é gerado; quando a célula sai do campo de visão, ela é removida da memória e seus recursos são liberados. As células são posicionadas em loops quadrados ao redor do usuário e são consideradas pertencentes ao campo de visão se estão a uma certa distância do usuário e dentro de um ângulo de 120º de visão. A figura 2.3 ilustra o funcionamento do campo de visão descrito e a figura 2.4 mostra uma cidade virtual gerada.

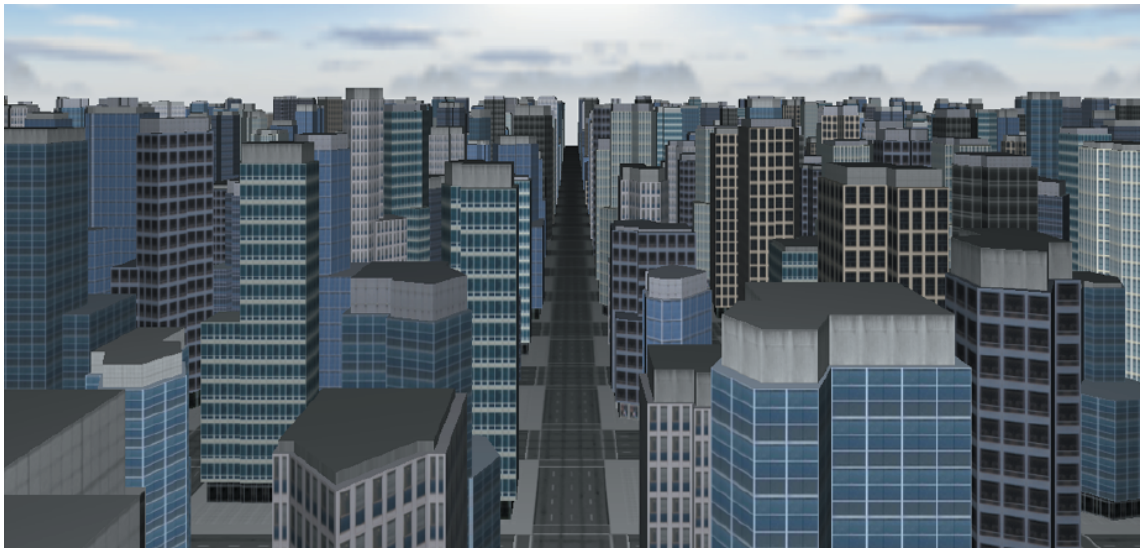

Figura 2.4: Exemplo de cidade virtual gerada [7]

Outro trabalho analisado foi o de [1] para a construção da ferramenta SkyCastle [3], uma *engine* para jogos online multijogador com suporte à geração de mundos virtuais gerados proceduralmente. No referido trabalho, o autor foca o problema de mundos virtuais cada vez maiores em jogos de computador e aplicações, o que cria a necessidade de desenvolvimento de ferramentas capazes de ajudar na geração de conteúdos realistas para esses mundos. Para a geração de relevo, o autor utiliza procedimentos parametrizados e sistemas baseados em fractais em uma abordagem de camadas: a aplicação adiciona um mapa de ruído (com amplitude reduzida) ao mapa de altura base em cada iteração. Os mapas de ruído são pré-computados e criados através de funções de ruído de Perlin, abordagem semelhante ao método de geração de relevo por deposição de sedimentos, que será abordado em mais detalhes na seção **??**.

Depois do assunto de criação de relevo, o autor aborda a funcionalidade de texturização da malha gerada. Um dos métodos apresentados é a utilização de uma imagem com proporções muito grandes, que seria capaz de cobrir o mundo inteiro. Embora essa abordagem passa ser útil para cenários pequenos, ela não é viável para terrenos grandes ou mundos virtuais, uma vez que o tamanho da imagem poderia atingir proporções proibitivas. Para contornar esse problema, a abordagem de reticulados é sugerida [5]; nessa abordagem, uma célula de textura é criada de tal forma que ao posicionar várias células, uma do lado da outra, o plano gerado apresenta uma texturização contínua e sem falhas. O resultado obtido com essa abordagem é aceitável, porém ele não é visualmente atraente para o usuário final, uma vez que o terreno apresenta uma continuidade que não existiria

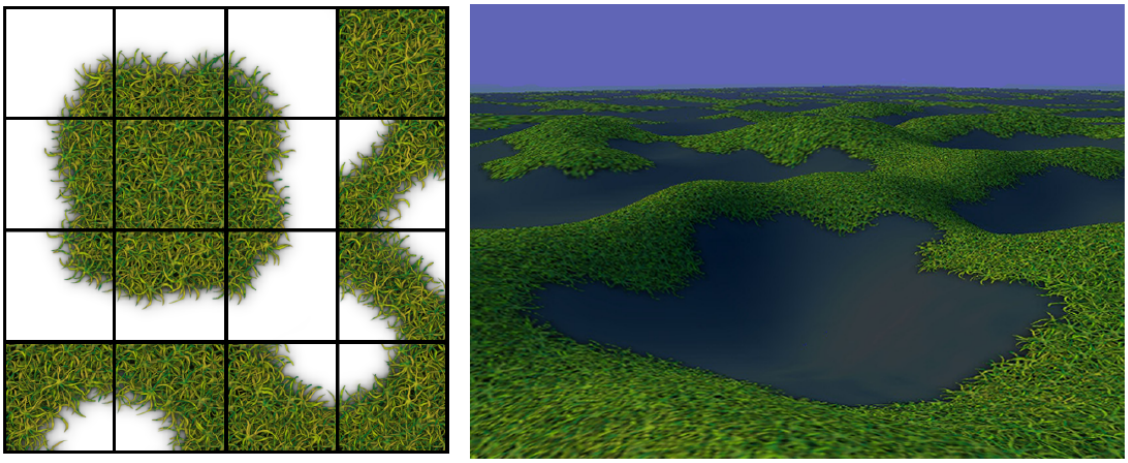

Figura 2.5: Terreno texturizado com duas camadas: a inferior sendo um bloco escuro e a superior sendo um conjunto de texturas em formato de borda com transparência [1]

no mundo real. Para conseguir um resultado visual melhor, o autor sugere a utilização de uma abordagem proposta por [6], que consiste na utilização de um conjunto de texturas pré-definidas em formato de borda juntamente com a texturização em reticulados já citada. O funcionamento do algoritmo resume-se em aplicar as texturas em borda sobre um plano já uniformemente texturizado, porém utilizando transparência nas áreas não desenháveis das texturas em borda. Dessa forma, a sobreposição das bordas sobre o plano texturizado irá produzir uma paisagem menos homogênea, o que gera um resultado visual melhor. A figura 2.5 ilustra as texturas em forma de borda e a sua utilização na abordagem descrita.

Para ornamentar o mundo virtual, o autor cita a utilização de plantas proceduralmente geradas através de três métodos principais: *L-System* [9], geração baseada em componentes [4] e árvores parametrizadas [8]. O algoritmo *L-System* consiste na geração de elementos a partir da interpretação de uma cadeia de caracteres, sendo que cada um desses caracteres representa uma estrutura geométrica ou operação (rotação, translação, etc); a cadeia de caracteres resultante é obtida a partir da aplicação sucessiva de regras sobre uma cadeia base. O algorítimo de geração baseada em componentes consiste na interpretação de uma árvore de elementos, sendo cada um desses elementos modificável através de parâmetros; cada componente pode possuir filhos e, também, uma descrição de qual é o elemento que pode ser usado como folha da árvore. Por fim, a geração de árvores parametrizadas é conceitualmente semelhantes à geração por componentes, exceto que a geração é orientada a ramificações; cada ramificação da árvore corresponde à

um nível de recursão, sendo o tronco da árvore o nível zero; quando um ramo sofre uma divisão, os filhos resultantes dessa divisão herdam algumas características do pai (como resolução), porém eles adquirem características próprias, como o ângulo de curvatura; ao final do processo, através de um estudo dos parâmetros corretos a serem utilizados (como a quantidade de ramos, o nível de recursão, etc), é possível que uma árvore completa seja gerada.

Outro trabalho analisado é a geração de planetas procedurais através de fractais e combinações deles com outros métodos [2]. Partindo da subdivisão recursiva de um octaedro, a autora cria um mundo esférico que serve como base para a aplicação dos algorítimos de relevo. Através de uma interface, o usuário pode estipular qual algorítimo de relevo ele deseja utilizar, e o resultado desse processo é traduzido em uma lista encadeada que descreve todos os vértices do mundo. Mais informações sobre algorítimos para geração de relevo podem ser encontradas na seção **??**). Uma lista encadeada foi utilizada para garantir que a malha resultante possa ter seus detalhes aumentados e/ou reduzidos facilmente (através das funções de inserção e remoção de elementos da lista), mesmo que uma lista encadeada não seja a melhor estrutura de dados para busca de um elemento em específico. Depois que a malha é gerada, a autora utiliza uma combinação de técnicas para colorir os pontos da falha; as técnicas variam conforme o resultado desejado, porém todas elas são baseadas na interpolação de cores parametrizada pela altura do ponto sendo analisado. Para garantir um aspecto mais real, utiliza-se em algumas técnicas conjuntamente com ruídos e turbulência, como o ruído de Perlin.

A figura 2.6 ilustra o relevo obtido com essa abordagem.

2.2 Funções de ruído

– Como utilizamos muito funções de ruido e afins, falar aqui do Perlin e do Musgrave. Falar bastante do livro deles, porque é uma coisa que é bem relacionada com o nosso trabalho e que a gente vai utilizar bastante.

– Só utilizamos o noise do Perlin aqui e o conceito de multi-fractal, mas sobre fractais eu vou falar depois.

2.3 Geração de relevo

sec:relevo

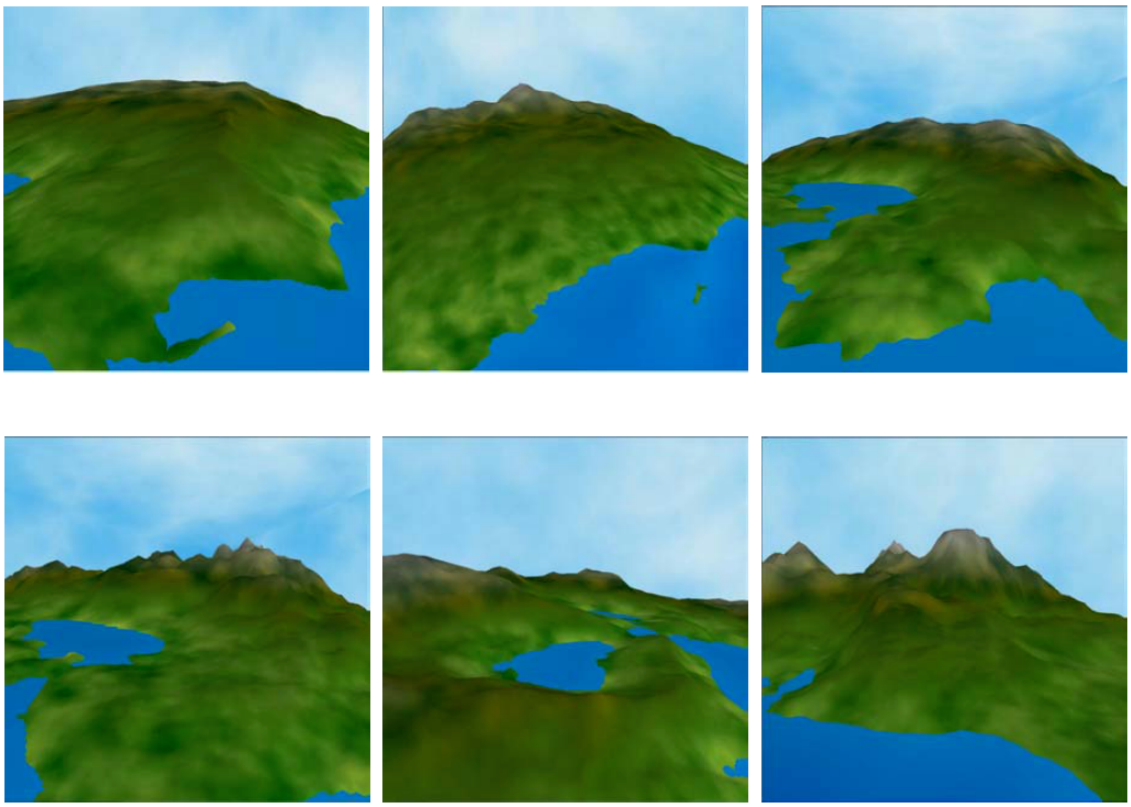

Figura 2.6: Diferentes relevos gerados pela utilização de fractais e suas combinações [2]

– Falar aqui das 3 formas que encontramos para geração de relevo através de fractais (deposição de sedimentos, alteração do ponto médio e divisão de não sei o que).

– Falar também sobre os multifractais que tem no livro do Musgrave, que ele usa para geração de costas de continentes muito bonitas.

3 IMPLEMENTAÇÃO

A ideia original do trabalho foi desenvolver um mundo virtual completo, semelhante em grande parte com o mundo real. Dentre as funcionalidades previstas, encontravam-se a divisão do terreno em relevos específicos (desertos, florestas, planícies, etc), cidades/vilas, caminhos entre as cidades, rios e cadeias montanhosas. A combinação de todos esses elementos seria capaz de criar um mundo virtual muito próximo da realidade, fato que seria de suma importância para garantir um bom resultado da ferramenta.

Quando o planejamento foi finalizado, a complexidade de determinadas funcionalidades previstas tornou proibitiva a sua implementação. A grande maioria dos problemas encontrados é uma consequência da abordagem de geração dinâmica de conteúdo sob demanda (a medida que o usuário se move, novos elementos são colocados na tela). A Figura 3.1 ilustra o problema da geração de conteúdo sob demanda.

Partindo do fato que o usuário só consegue enxergar aquilo que está dentro do seu campo de visão, todos os algoritmos de geração de conteúdo, seja para relevo, caminhos ou cidades, precisam levar em consideração única e exclusivamente as informações que estão disponíveis dentro desse campo. Essa abordagem é eficiente para a utilização racional de recursos (processar somente o que o usuário está vendo), porém ela aumenta

Figura 3.1: Problema da geração de conteúdo sob demanda.

a complexidade dos algoritmos envolvidos na ferramenta.

Para o algoritmo de geração de cadeias montanhosas, por exemplo, não é possível determinar onde a cadeia termina, visto que o mundo fora do campo de visão tecnicamente não existe ainda, ele será gerado conforme o usuário avança pelo terreno. Uma abordagem seria utilizar uma função matemática que descrevesse a cadeia montanhosa, porém essa função não deveria depender de um ponto de início e fim, porque eles poderiam inexistir em um determinado momento. Se a função de geração de cadeias montanhosas não dependesse de um ponto de início e fim, ela precisaria, ao menos, depender da posição do usuário no mundo virtual para que o conteúdo correto fosse gerado. Depender de uma localização implicaria que a cadeia montanhosa gerada pela função fosse pré-posicionada no mundo virtual, o que iria contra o conceito de geração de conteúdo sob demanda.

Além disso, os algoritmos são sensivelmente afetados pelo fato de que as informações que eles recebem em um determinado instante podem desaparecer por completo na próxima iteração, visto que o usuário pode se mover e mudar o conteúdo do campo de visão. Utilizando o exemplo da geração de cadeias montanhosas, uma montanha poderia sofrer uma alteração em sua composição de forma abrupta, apenas porque os pontos que estavam sendo utilizados para a geração do relevo mudaram.

Para contornar esses problemas e focar os esforços de desenvolvimento em soluções pontuais, a geração do mundo virtual foi dividia em três grandes etapas: terreno infinito, continentes e relevo. A geração de conteúdo sob demanda afeta de forma diferenciada cada uma dessas etapas e a descrição da implementação de cada uma delas, junto com os problemas associados, é descrito nas seções seguintes.

3.1 Terreno infinito

A base para a geração do mundo virtual proposto é a possibilidade do usuário poder andar, de forma infinita, sobre a superfície do mundo e, conforme anda, visualizar novos conteúdos. A medida que o usuário anda, a ferramenta precisa ser capaz de identificar em qual local do mundo o observador se encontra para então gerar os conteúdos à sua volta.

Para solucionar esse problema, utilizou-se uma variação da técnica descrita por [7]. Na abordagem dos autores em questão, o mundo virtual pseudo-infinito é divido em células quadradas e, à medida que o usuário anda, as células são adicionadas e/ou removidas do campo de visão. Cada célula possui um conteúdo próprio e auto-contido, ou seja, a

Figura 3.2: Organização do sistema de coordenadas do mundo virtual

célula não precisa de informações de vizinhos para gerar o seu conteúdo. Isso garante que as células não entrem em uma dependência recursiva infinita entre elas para conseguirem gerar o seu conteúdo. Além disso, essa abordagem é vantajosa para garantir o uso racional de recursos, uma vez que só serão carregados para a memória os blocos que o usuário realmente consegue ver.

Para o posicionamento do usuário no mundo virtual, os autores utilizam um vetor 3D no formato `(x,y,z)`. Conforme o usuário se move horizontalmente pelo mundo, as coordenadas `x` e `y` são atualizadas. Se o usuário se move verticalmente, a coordenada Z é alterada. A abordagem utilizada para a ferramenta dessa dissertação baseou-se nesses conteitos. A figura 3.2 ilustra a organização do mundo virtual.

A origem do mundo virtual é o ponto `(0,0,0)` e os eixos que definem o plano horizontal são o `X` e `Z`, sendo a altura controlada pelo eixo `Y`. A distância máxima que o usuário consegue percorrer em qualquer um dos eixos é o número máximo suportado por um inteiro de 32 bits com sinal.

O que o jogador consegue ver na tela em um determinado momento é um pedaço do mundo virtual existente. Esse pedaço foi chamado de *view frustum*, ou campo de visão. Diferentemente do que foi feito em [7], no qual o campo de visão é um cone, o campo de visão da presente ferramenta é um quadrado centrado no usuário. A figura 3.3 ilustra o funcionamento do campo de visão.

A partir da posição `(x, y, z)` do usuário, a ferramenta calcula qual é o conteúdo visualizável ao redor do referido ponto. Ao chegar na borda limite do mundo, que pode ser a distância máxima de um eixo, por exemplo, o usuário é impedido de avançar e nenhum conteúdo é mostrado além da borda limite.

Inicialmente planejou-se a utilização de quadrados para dividir o mundo virtual em

Figura 3.3: Campo de visão do usuário: a área visualizada corresponde a uma fatia do mundo virtual existente.

células, conforme é feito na outra abordagem citada anteriormente. A utilização de células garantiria que o mundo virtual fosse subdivido em blocos menores, o que viabilizaria um melhor controle sobre o que o usuário consegue ver no seu campo de visão e, também, um melhor controle sobre a geração de conteúdo. O maior problema encontrado nessa abordagem, que foi a razão pela qual ela foi abandonada, é a dependência que as células precisam ter entre si para que o terreno infinito seja gerado.

Utilizando o exemplo da geração de relevo, que é tratada em mais detalhes na seção 3.3, se o mundo virtual fosse dividido em células, cada uma delas deveria possuir um relevo perfeitamente nivelado com a célula vizinha, caso contrário o relevo gerado teria diversos "degraus". Analisando o problema um pouco mais a fundo, no caso da geração de montanhas, por exemplo, se na metade de uma célula a ferramenta decidisse que uma cadeia montanhosa deveria começar, a célula vizinha deveria obrigatoriamente ter a continuação dessa cadeia montanhosa, caso contrário a montanha em questão seria fatiada pela metade. Uma das formas de corrigir esse problema é adicionar um nível mínimo de dependência entre as células: o conteúdo de uma célula é gerado com base nas informações da própria célula e também com base em alguma "dica"da célula vizinha. No exemplo da cadeia montanhosa, a célula vizinha ao começo da cadeia saberia que o conteúdo que ela deve gerar é a continuação da cadeia montanhosa, visto que a sua célula vizinha possui o começo da cadeia.

Essa dependência de conteúdo gera um encadeamento recursivo infinito entre as células. Se a célula `A`, por exemplo, for gerar o seu conteúdo, ela irá fazer isso com base nas suas informações e também com base nas informações de sua célula vizinha, `B`. A célula `B`, por sua vez, só poderá informar a `A` o seu próprio relevo quando ela o gerar; para gerar

Figura 3.4: Mapeamento das coordenadas do mundo virtual para as coordenadas de desenho da tela

o seu relevo, ela precisa das suas informações e das informações da sua vizinha, `C`, e `C` precisa de `D` e assim por diante. Dessa forma, para gerar o conteúdo de `A`, a ferramenta teria que obrigatoriamente percorrer todas as células do mundo virtual.

Uma saída alternativa para esse problema da recursão infinita é definir um nível de consulta de informações. Embora essa abordagem limite o nível de consulta entre as células vizinhas, ela não soluciona o problema de continuidade de conteúdo. Se o conteúdo de `A` for gerado com seis níveis de recursão, por exemplo, quando o usuário se mover, novas células vizinhas serão consultadas para a geração do conteúdo; se a última célula que `A` consultou foi `G`, depois que o usuário se mover, `G` terá novas informações para o seu relevo, porque agora ela pode solicitar informações de suas vizinhas. Isso fará com que todas as células dependentes de `G` mudem o seu conteúdo, o que resultaria em um relevo diferente a cada movimentação do usuário. Em virtude da complexidade descrita e dos problemas mapeados, a abordagem de divisão do mundo virtual em células foi abandonada e substituída pelo modelo de campo de visão quadrado.

3.1.1 Renderização de conteúdo

Depois que o modelo de controle da geração de terrenos infinitos foi definido, iniciaram-se os trabalhos de computação gráfica para que as informações pudessem ser desenhadas na tela. Embora o mundo virtual tenha coordenadas pseudo-infinitas, o máximo de conteúdo que o usuário enxerga na tela é a área do campo de visão, que possui sempre o mesmo tamanho e coordenadas de desenho. A figura 3.4, quadrado 1, ilustra as coordenadas envolvidas no desenho do conteúdo do campo de visão.

O plano `A` representa o mundo virtual gerado pela ferramenta, enquanto o plano `B` representa a fatia do mundo virtual que o usuário consegue visualizar. A medida que

o usuário se move, o centro do campo de visão é alterado e o usuário passa a ver novos conteúdos. Independente da movimentação que o usuário faça, o plano `B` pode sempre ser mapeado como se estivesse na origem, porque ele é apenas uma fatia do mundo virtual. O que a ferramenta faz para desenhar esse conteúdo na tela é extrair essa faria e, então, mapeá-la para um mapa de altura (*height map*). Depois que o mapa de altura é definido, as coordenadas dos eixos `X` e `Z` do mundo virtual que foram utilizadas durante a extração não são mais relevantes para o processo de desenho. Dessa forma, depois de extraído, o mapa de altura possui sempre as mesmas coordenadas nos eixos `X` e `Z`, que são as coordenadas de desenho da tela. A única informação do mundo virtual que é mantida é a altura de cada um dos pontos da malha do mapa de altura, conforme ilustra a figura 3.4, quadrado 2. O mapa de altura gerado é renderizado como uma malha triangular.

3.2 Continentes

– falar que utilizamos o algoritmo maluco do professor dos EUA para fazer a costa. Falar que existe o mundo é gigante o suficiente para fazer com que uma matrix que descreve água/terra seria inviável. Para solucionar esse problema, falar que utilizamos o conceito de isLang local e isLang global. O isLand global dá uma dica se o lugar é água ou terra, e o isLand local utiliza multifractais para fazer o desenho das bordas dos continentes.

– É importante frisar que essa seção é onde está a nossa contribuição na pesquisa: mundo pseudo-infinito, com relevo gerado on-the-fly e com costas de continentes com multiresolução.

3.3 Relevo

A ideia original do presente trabalho foi desenvolver um mundo virtual capaz de apresentar diversos tipos de relevos, como cadeias montanhosas, planícies, vales, desertos, florestas, etc. A complexidade associada à geração de cada um desses elementos varia conforme o nível de realismo esperado, bem como pelo nível de dinamismo do conteúdo gerado. Quanto mais presente for o conceito de geração de conteúdo sob demanda, mais difícil torna-se a tarefa de gerar conteúdos conexos e sem falhas abruptas na malha de relevo.

Mantendo-se a meta de gerar o conteúdo da forma mais dinâmica possível (sem ele-

Figura 3.5: Geração de conteúdo com base apenas as informações do campo de visão

mentos pré-posicionados, por exemplo), o primeiro grande problema encontrado durante o desenvolvimento do relevo do mundo virtual foi a forma como ele deveria ser gerado. Considerando que a ferramenta é capaz de criar um mundo pseudo-infinito, a geração de uma malha de relevo tão grande, de uma só vez, é fisicamente inviável; a quantidade de vértices que precisariam ser armazenados tornaria o consumo de armazenamento da aplicação muito grande, o que poderia ser um empecilho para a utilização da ferramenta.

Além do problema de armazenamento, outro tópico importante que foi considerado é a geração de conteúdo contínuo, ou seja, um relevo que não tenha falhas abruptas de continuidade, como uma cadeia montanhosa que termina inesperadamente. Partindo do fato que o mundo virtual não poderia ser divido em células, tendo em vista os diversos problemas de continuidade de conteúdo, a solução para a geração de relevo deveria basear-se apenas nas informações disponíveis no campo de visão. Embora seja possível gerar relevo utilizando-se como valores de entrada somente as informações do campo de visão, novos problemas de continuidade foram encontrados.

Supondo que a geração de relevo utilize a posição do vértice no mundo virtual como semente para uma função pseudo-aleatória de cálculo de relevo; supondo também que a geração de montanhas, por exemplo, é definida por pontos chave que indicam a espinha dorsal da cadeia montanhosa. Se o algoritmo se basear apenas nesses pontos chave para calcular as montanhas, o usuário só irá enxergar a montanha quando um desses pontos chave entrar dentro do campo de visão. Isso irá causar falhas abruptas de continuidade, porque se o usuário costear a espinha dorsal da montanha, sem que os pontos chave entrem no campo de visão, ele não verá a cadeia de montanhas, sendo que ele deveria ver um relevo ascendente até o cume dessas montanhas. A figura 3.5 ilustra esse problema.

A solução encontrada para a geração de relevo foi utilizar uma função paramétrica

Figura 3.6: Função de geração de relevo parametrizável e o resultado gerado

que informa a característica de cada um dos vértices do mundo virtual. Nessa abordagem, cada ponto do mundo é utilizado como parâmetro para a função de relevo que, por sua vez, calcula a altura que esse ponto deve ter. Como resultado, tem-se uma função capaz de descrever todo o relevo do mundo virtual, independente do tamanho que ele seja; a quantidade de recursos computacionais que serão utilizados para gerar o conteúdo do campo de visão é diretamente proporcional ao tamanho do campo de visão, não ao tamanho do mundo, visto que a função de relevo utiliza as informações de cada ponto para calcular a sua característica. A figura 3.6 ilustra o funcionamento da função de relevo aplicada ao mundo virtual gerado.

Para aumentar a heterogeneidade do relevo gerado, a ferramenta utiliza duas funções de relevo, uma para o eixo `X` e outra para o eixo `Z`. Além de garantir que o relevo gerado não seja perfeitamente simétrico nos dois eixos, essa abordagem permite um maior controle sobre a geração de conteúdo. A altura de cada ponto do mapa é definido pela soma dessas duas funções de ruído.

Inicialmente utilizou-se uma combinação de funções seno e cosseno para a geração do relevo.

4 RESULTADOS

5 CONCLUSÃO E TRABALHOS FUTUROS

–Trabalhos futuros: – Melhoramento nas funções que geram relevo; – Texturização conforme altura do terreno e afins; – Rios; – GPU;

REFERÊNCIAS

[1] A. Aho and J. D. Ullman. *Compiladores: conceitos, técnicas e implementaćão.* Makron Books, 1993.

[2] A. Aho and J. D. Ullman. *Compiladores: conceitos, técnicas e implementaćão.* Makron Books, 1993.

[3] A. Aho and J. D. Ullman. Compiladores: conceitos, técnicas e implementaćão., 1993.

[4] Bernd Lintermann and Oliver Deussen. A modelling method and user interface for creating plants, 1998. Computer Graphics Forum.

[5] Filip Strugar. Advantage terrain, 2007. balbal.

[6] Filip Strugar. Advantage terrain, 2007. blahblah.

[7] Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Realtime procedural generation of 'pseudo infinite' cities, 2005.

[8] JasonWeber and Joseph Penn. Creation and rendering of realistic trees, 1995. blabla.

[9] Prusinkiewicz Przemyslaw and Aristid Lindenmayer. The algorithmic beauty of plants, 1990. Springer-Verlag.

[10] Thomas Wang. Integer hash function. Technical report, 2000. Available at: <http://www.concentric.net/ Ttwang/tech/inthash.htm>.