

# Charack: ferramenta para geração em tempo real de mundos virtuais pseudo-infinitos para jogos 3D

Fernando Bevilacqua

Cesar Tadeu Pozzer

Universidade Federal de Santa Maria

## Abstract

Em jogos *massively multiplayer online* (MMO) a existência de um mundo virtual persistente é um tópico importante para manter o jogo atrativo e divertido ao jogador. Quanto maior o mundo a ser explorado, tecnicamente mais tempo o usuário passará jogando para conseguir explorar o maior número possível de lugares. A solução proposta neste trabalho é uma ferramenta capaz de gerar mundos virtuais pseudo-infinitos com diversificação de formas e relevos ao longo de sua extensão. Utilizando uma combinação de algoritmos e métodos de gerenciamento de conteúdo, a ferramenta é capaz de criar ilhas/arquipélagos, baías e costas contendo falésias, praias e rochedos que imitam as paisagens encontradas na natureza. Dentre as inovações apresentadas, estão a criação de um terreno virtual de vastas proporções com enfoque mais detalhado no que diz respeito à geração da costa.

**Keywords::** MMO, mundos virtuais, geração de terreno, jogos 3D, ruído, geração procedural, multifractal

## Author's Contact:

{fernando,pozzer}@inf.ufsm.br

## 1 Introduction

## 2 Introdução

The computer games market has been evolving considerably over the years. Since the first console, the hardware performance has increased and new graphic technologies were developed, resulting in a wide range of themes and game styles. In the multiplayer games, players interact with other human beings and also with NPCs, which are represented by virtual characters. The kind of game is popular and the social interaction between players is a matter of research [Griffiths et al. 2003; Ducheneaut et al. 2006]. In the category of multiplayer games there are the massively multiplayer online (MMO) ones, which are online games featuring large number of players interacting with each other in a persistent virtual world.

An MMO can feature millions of players, such as EverQuest [Sony Entertainment 2007] and World of Warcraft [Blizzard Entertainment 2007], the latter with more than 6 million subscribers [Clark 2006]. A persistent virtual world is an important topic to keep the game fun and attractive to the player. The bigger is the world to be explored, technically the bigger is the time the player has to spend in order to explore all the places. As a result of such huge virtual worlds, the creation (and subsequent upgrade) of those worlds is a complex task. EverQuest and World of Warcraft present a virtual world with a wide diversity of geographical features such as mountains, valleys, forests, fields, caves, etc., and the vast majority of them nominated and related the history of the game. The manual creation of those virtual worlds requires a team able to create heightmaps, adorn landscapes, ensure usability of the map (avoid unreachable places, for example), create interesting place for players, etc. To help on that task, the development of a tool able to generate complex virtual worlds is useful to speed up the development of 3D games such as MMOs.

The solution proposed in this work is a tool able to generate complex virtual worlds in real-time using noise-based techniques of terrain generation. The proposed tool is able to generate complete virtual world featuring continents, oceans, flat areas, mountains, etc,

with all those elements generated in real time, on demand as the player moves along the world.

## 3 Overview

The tool was designed to allow developers to use its features in order to generate 3D terrains for games, particularly MMOs, with minimal human intervention in the generation process. The size of the resulting world is customizable, but it can reach billions of pixels. The use of the term "pseudo" is necessary due physical limitations in computers hardware: an unsigned integer, for instance, can store a certain amount of data; if there were no physical limitations, the tool would be able to generate, in fact, an infinite world.

The content generation is made on demand. As the user moves along the world, the elements inside the user's view are processed and stored into the memory and the ones away from the user's view are removed. Even though the generation of all elements is based on random calculations, if the player visits a specific point A, then walks for miles generating a completely different set of landscapes, and returns to point A, the same previously seen landscape will be shown again.

Besides the heightmap generation, the tool is also able to create oceans and continents, all of them customizable. Unlike other related works which deal with the generation of continents as a consequence of the heightmap generation, this paper presents as its main contribution a new approach for content generation in virtual worlds. In this approach, the generation of continents, topography and coastlines are handled separately.

The shape of continents is defined on a macro level as a result of an algorithm for maps generation that work projecting pixels onto a sphere. The coast line generation is performed on a micro level. As a result, the tool is able to produce cliffs, beaches, bays and islands.

## 4 Related works

### 4.1 Virtual city

The procedural content generation is an old topic in the field of computer graphics. The application of such technique to generate a complete virtual world was used by [Greuter et al. 2005], whose goal was to create a virtual city that was visually interesting and complex.

In the approach the world were divided into a grid consisting of several squares, called cells. The location of each cell is used with a global seed as an input for a hash function [Wang 2000]. The result of this function is used as a seed for a pseudo random number generator and it will define all the characteristics of the buildings within a cell. As a consequence of that approach the contents of a cell is always the same, no matter how far the user moves or if the cell is removed from memory.

To ensure acceptable use of memory and CPU, only the elements inside the user's view are generated. As the user moves through the city, new cells are added to the user's view and the new content is generated. When the cell leaves the user's view, it is removed from memory and its resources are released. The cells are placed in square loops around the user and are considered inside the user's view if it is close enough and if it is located within a 120° view range.

## 4.2 Layered generation and texturization

Another related work was a tool used to build the SkyCastle multiplayer game engine [Häggström 2006; Häggström 2009]. For the heightmap generation, parameterized procedures and fractal based systems are combined in layered approach: starting with a base map, the application merges a new map with the base in each iteration. The new maps are pre-calculated and generated using Perlin noise.

One of the texturization methods presented are the use of a large image able to cover the whole world. Although this approach is useful for small scenes, it is not suitable for large scenes nor virtual worlds, since the image size could reach prohibitive proportions. To avoid this problem, an reticulate approach is suggested [Cohen et al. 2003]; using that approach a texture cell is created so that the combination of them produces a smooth and continuous and textured plane. The results are acceptable, but it is not visually attractive to the user, because the landscape features an unusual pattern that will never exist in the real world. In order to achieve a better visual result, an approach proposed by [Lefebvre and Neyret 2003] is suggested. In that approach, a set of pre-defined border textures is combined with reticulate textures. The idea is to apply the border textures on an already textured plane, but using transparency in the border textures. As a consequence, border textures will overlap the some regions of the plane and produce a less homogenous landscape, which creates a better visual result.

In order to decorate the virtual world, three procedural plant generation methods are used: L-System [Przemyslaw and Lindenmayer 1990], component-based generation [Lintermann and Deussen 1998] and parameterized trees [JasonWeber and Penn 1995].

## 4.3 Procedural planets

The use of fractals and their combinations with other methods are also used to procedurally generate planets [Linda 2007]. In this approach, the generated world is not infinite, but it is spherical and simulates the view of the planet Earth. Starting with a recursive subdivision of an octahedron, a spherical world is created and it is used as a first step to sevel algorithms of heighmap generation. The heightmap is obtained as a result of several techniques, each one featuring its peculiarities and results. Among the techniques used are: generation using random failures, midpoint displacement (including a multifractal variation) and Perlin noise (and its many variations).

## 4.4 Erosion fractals

Another approach for procedural terrain generation in real time uses fractals affected by erosion [Olsen 2004]. As pointed out, the increasing processing power of home computers has enabled games to make use erosion simulations almost in real time. As a consequence, an approach envolving fractals and erosion is possible to be reached, resulting in more realistic terrain. In order to achieve that, a heightmap was created from a Voronoi diagram, where each vertices of the diagram is called *functionality point*; the value of each cell in the heightmap is obtained through a linear combination of distances of the closest functionality points. Once the heightmap is generated, it is combined with a noise algorithm in order to produce mountains. The sharp lines introduced by the well-defined Voronoi diagram are removed with the application of a turbulence filter. For the erosion simulation an hybrid method is used, which is a combination of two techniques: the thermal [Musgrave et al. 1989] and the hydraulic erosion.

## 4.5 Stochastic subdivision of a quadtree

Another related work uses stochastic subdivision techniques in order to generate a virtual world [Dollins 2002]. The main goal was to create a virtual world of large proportions, however the creation of its content is procedurally made and performed on-demand with a multi-resolution approach. For the heightmap generation a recursive sub-division process is applied to a quadtree. The height of each vertex is defined by the midpoint of each cell in the quadtree.

For each level of subdivision, the midpoint of the each new child cell is defined by the parent cells around the cell being divided. The height of all other vertexes, such as the one at the corners, is defined as a result of an interpolation of the closest midpoints. In each level of detail, more cells are divided and replaced with the new generated child cells.

## 4.6 Map generator

Another related work focusing on the generation of maps for planets, which is based on the generation of spheric worlds using a recursive subdivision of a tetrahedron [Mogensen 2009]. Although the work is not presented as a graphical tool where the user can explore a 3D environment, all generated information is part of a complete virtual world featuring highly customizable continents and oceans. The presented idea is to create a map as a result of a projection of pixels onto a sphere, a method similar to ray tracing [Glassner 1989]. In the same way it is performed in ray tracing, where rays leaving the camera collide with something and create a visible pixel, each pixel of the map being generated is projected onto sphere. The projection algorithm sets a height value to each pixel found, resulting in a complete heightmap. In order to render the map, initially the visible points on the sphere surface are found. After that, the height value for each one of those points is calculated as follows:

- Insert the sphere inside the tetrahedron.
- Divide tetrahedron into two smaller tetrahedral.
- Choose which of the two tetrahedral contains the point being analyzed.
- Repeat the above steps until the tetrahedron is small enough.
- Use the average height of the each tetrahedron vertex in order to calculate the height of the point being analyzed.

Initially the sphere is placed inside a irregular tetrahedron and information about height and a random seed are set for each of the vertexes. After the tetrahedron is divided in two pieces by a plane formed between the midpoint of the longest edge and the two end points of the edge in front of the midpoint of the longest edge. In short, each of the two resulting tetrahedral have three points from the original tetrahedron and a new vertex placed in the midpoint of the largest edge. The height and the seed of the new vertex is calculated from the height and seed of the end points of longest edge. The process is repeated recursively for each of the tetrahedral until it is small enough to contain only the desired point.

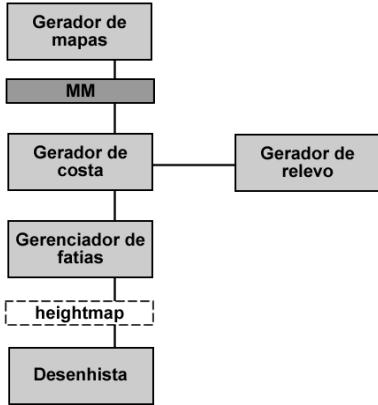
# 5 Tool organization

The following sections explain in a top-down approach how the tool was structured to achieve the described goals.

## 5.1 Initial analysis

There are several related works concerning the generation of virtual worlds, whether finite or infinite. One of the realted works creates an infinite city [Greuter et al. 2005] that is presented to the user on demand as it walks on the ground. This work was the ground zero for Charack development, however the original idea was changed in order to make the tool suitable to generate more types of terrains (mountains, plains, continents, etc.), not only streets and buildings. The approach of content generation made on demand was maintained.

The procedurally generated world approach [Linda 2007] is very close to the concept of content generation aimed for Charack. In that work, a spherical planet is created as a result of a recursive division of a geometric shape, then noise functions are applied to the mesh to generate the heightmap. There is no distinction between the content generation approache for continents and the content generation for the lands within the continents. As a result the continents are created by flooding the heightmap with a water plane, which will produce the coast lines based on height of the sea level and the



**Figure 1:** Basic structure for content generation

amount of ripples in the topography. The content itself is not generated on demand. Charack was created from an evolution of this idea, but subject to some limitations. The world created by Charack handles differently the generation of continents, coast lines and the heighmap within them. Each of these generation processes has its own peculiarities, which can be tweaked in order to customize specific details in the world with no interference in the other processes. This is the case of the creation of long beach on the coast without changing the basic structure or the topography of the continent that the beach is inserted. Charack also generates the content on demand, however it doesn't use a spherical approach, the resulting world contained within a plan.

The generation of a virtual world as a result of recursive subdivisions of a quadtree [Dollins 2002] is very similar to the Charack proposal. In that work, a world with huge proportions is created and its content is generated on demand as the user moves. The heighmap is created in a parameterized and multi-resolution way, so the closer the user of place, the greater is the amount of detail there. There is also no distinction in the generation process of continents/coastlines/land content. The heighmap generation proposed is used by Charack, however continents and the coastlines generation process are completely different.

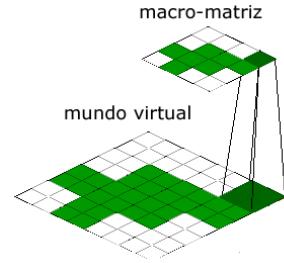
Analyzing the related work, virtual worlds are generated several approaches, but in none of them handles differently the content generation for continents, coastlines and topography. Although there are variations in how the heighmap is created, the generation of continents is a result of a water flooding plane. This approach allows the developers to focus on the content generation for the land, however it has a simple approach concerning continents and coastlines. The main idea and contribution of Charack is the content generation handled differently for each world element (continent, coastline, etc.), with an aggressive and specific approaches for each one. Most of the related work focuses on heighmap generation, so we believe the Charack contribution will be better if the efforts are focused on the generation of continents and coastlines.

## 5.2 Basic structure

In order to create a virtual world that reaches the presented proposal, a top-down approach is used for the content generation. The Charack data flow begins in a macro view of the world, which are the continents, evolving to a micro view of the planet, which are the generation of content for each vertex that will be drawn in the screen. Figure 1 shows Charack basic structure.

### 5.2.1 Maps generator

At the top of the chain is the maps generator, which creates the continents that exist throughout the virtual world. This module is an encapsulation of the solution created by [Mogensen 2009], which was described in the section 4.6. When the tool is initiated, it uses a user defined seed to generate all the continents. Once the continents are generated, all the information related to terrain types



**Figure 2:** Mapping the MM to the virtual world: each MM pixel represents several pixels in the virtual world.

(land, water and coast) are stored in a matrix, called **macro-matrix** (**MM**), which is used by all the other algorithms.

### 5.2.2 Slice manager

Below the MM and the maps generator is the slice manager (SM). It extracts a portion of the virtual world (the user's view) and provide the render engine with information about the heightmap. In order to obtain the required information to create the heightmap, the slice manager uses the coastline generator (CG), which uses height generator (HG) and the data stored in the MM.

In the context of the SM, there is no information about land or water, it only knows a set of pixels of the virtual world and its height values. Using the position of the user as a guide, the SM slices virtual world and, for each collected pixels, it queries the CG in order to find out its height value.

### 5.2.3 Height generator

The height generator (HG) defines the height value for each pixel in the virtual world. To ensure that the developers can create a customizable heightmap based on their needs, new functions to generate content can be added to the tool in a simple way.

### 5.2.4 Coastline generator

The coastline generator (CG) will map each pixel of the slice manager to the MM in order to find out the terrain type of that pixel. If the pixel being analyzed is mapped to a location in the MM that is described as water, then the CG assigns a height value equals to sea level for the pixel, after that returning that pixels to the SM. If the pixel is mapped to a place described as (simple) land, then the CG will use the information provided by the HG in order to find out the height value for that pixel. Finally, if the pixels is mapped to a place described as coast, then the CG uses its own structure (together with the MM) to set height value for the that pixel.

The resulting virtual world technically has height and width defined by the maximum size of a signed integer. It is physically impossible to generate a MM with such proportions. Since the MM is smaller than the virtual world, an MM's entry  $(i, j)$  represents several pixels in the virtual world. Figure 2 illustrates the MM mapping process.

As it can be seen in the figure the smaller is the MM, the more pixels in the virtual world will be represented by the same entry in the MM. If the virtual world had  $1000 \times 1000$  as its size and the MM had  $10 \times 10$  as its size, for instance, it means that for each pixel of the MM represents 100 pixels in the virtual world. If one of these pixels is described in the MM as coast, then there is an area of  $100 \times 100$  pixels in the virtual world that must be a coast. To solve this problem the CG acts. When any of those pixels in that particular area of the virtual world is analyzed by the CG, it will work on it and return it with different values, which will result in a coast line for that area, not an area entirely filled with land or water.

### 5.2.5 Rendering engine

The rendering engine draws created heightmap in the screen. The result is rendered as triangles mesh that is textured according to the height value of each vertex in the mesh.

## 6 Implementação

O principal problema encontrado durante a implementação de Charack foi a 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 a complexidade dos algoritmos envolvidos.

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 dividida 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.

### 6.1 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 [Greuter et al. 2005].

A solução encontrada foi fazer com que o jogador veja na tela em uma determinado momento um pedaço do mundo virtual existente, porém sem a utilização de células ou quadrantes. Diferentemente do que foi feito em [Greuter et al. 2005], no qual o campo de visão é um cone, o campo de visão de Charack é um quadrado centrado no usuário. A partir da posição ( $x$ ,  $y$ ,  $z$ ) do usuário, calculase qual é o conteúdo visualizável ao redor desse ponto e, então, "recorta-se" essa fatia do mundo e a desenha-se ela na tela. 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.

Depois que a fatia visível é recortada, ela é desenhada na tela. Para texturar o conteúdo renderizado, utiliza-se um conjunto de imagens gerenciadas através de GLSL [OpenGL 2007]. Baseado na altura do pixel sendo desenhado, calcula-se o peso que cada



**Figure 3:** Conjunto de imagens utilizadas para texturização do terreno

uma das texturas possui naquele local e, em seguida, cria-se uma interpolação delas. Cada uma das texturas do conjunto possuem um intervalo de altura na qual podem atuar: areia para alturas baixas, pedra para alturas médias, cascalho para alturas intermediárias e grama para as alturas maiores que as já estipuladas. Um exemplo de funcionamento desse método é a texturização da areia da praia; nesse caso, todos os pixels da praia possuem uma altura que faz com que a textura de areia tenha o peso máximo, ao passo que as demais alturas tenham um peso zero. A figura 3 demonstra o conjunto de texturas existentes.

### 6.2 Relevo

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 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. Os algoritmos de geração de relevo devem trabalhar apenas com as informações existentes no campo de visão, evitando comportamentos estranhos causados pela renovação de pontos dentro do campo de visão à medida que o usuário se move.

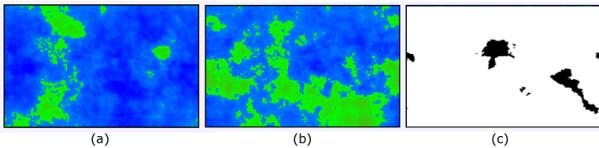
Para a ilustração do problema de geração de conteúdo contínuo, supõe-se o seguinte cenário: a geração de relevo utiliza a posição do vértice no mundo virtual como semente para uma função pseudo-aleatória de cálculo de relevo; supõe-se também que a geração de montanhas, por exemplo, é definida por pontos 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 solução encontrada para a geração de relevo foi utilizar uma função paramétrica 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 possua; 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.

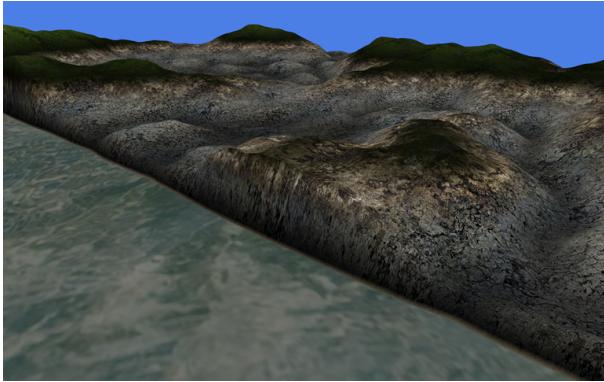
Para a criação da função de relevo, utilizou-se o método de geração de ruído de Perlin, que também foi utilizado por [Linda 2007]. Para garantir um controle sobre o nível de perturbação do relevo utilizou-se apenas as propriedades da função de ruído de Perlin; para a obtenção de relevos mais "enrugados", utilizou-se mais oitavas na função de ruído.

### 6.3 Continentes

A geração de continentes e oceanos foi proposta para quebrar a monotonia de uma paisagem composta apenas por terra no mundo



**Figure 4:** Planetas aleatórios gerados com diferentes sementes de entrada. (a) e (b) mapas com informações de relevo; (c) mapa com informações somente sobre o que é terra/mar.



**Figure 5:** Resultado gráfico obtido quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher as discrepâncias de mapeamento da MM

virtual, bem como para aumentar o nível de semelhança com as paisagens encontradas na natureza. A solução encontrada para garantir que os continentes sejam mais customizáveis foi pré-processar as faixas de terra existentes no mundo e, então, armazenar essa informação para os demais cálculos da ferramenta. Com essa abordagem, a geração de conteúdo sob-demanda foi parcialmente quebrada, uma vez que os continentes serão gerados por completo antes de todos os demais conteúdos, porém isso garante um melhor controle sobre o que é oceano e o que é terra.

A geração dos continentes é baseada na abordagem de [Mogensen 2009], descrita em detalhes na seção 4.6. Utilizou-se esse gerador de planetas porque ele possui diversas opções de parametrização, como estipulação da semente que será usada para os cálculos aleatórios, definição da altura/largura do mapa gerado, suavidade das curvas e zoom. A figura 4 (a), (b) e (c), ilustra os resultados obtidos com o gerador de planetas de [Mogensen 2009].

### 6.3.1 Problemas na geração de continentes

Conforme explicado na seção 5.2, cada pixel da MM é mapeado para diversos pixels no mundo virtual. Uma consequência direta desse mapeamento desproporcional é a geração de grandes faixas de terra retilíneas; se fosse possível a geração de uma matriz com o tamanho exato do mundo virtual, ela conteria a resolução necessária para a ferramenta decidir com precisão se um determinado pixel é ou não terra, em uma proporção de 1:1 (um pixel da MM corresponde a um pixel do mundo virtual). Essa abordagem, porém, não é viável, visto que uma matriz com tais proporções consumiria muitos recursos para ser construída e processada. Embora a ferramenta permita que o tamanho da MM seja ajustado para qualquer valor arbitrário, testes realizados mostraram que um tamanho de 800x800 é o bastante para que informações suficientes sejam processadas pelos demais algorítimos da ferramenta.

A figura 5 ilustra o resultado gráfico obtido pela ferramenta quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher os espaços retilíneos entre um mapeamento e outro.

Nessa ilustração está sendo mostrado um local no mundo virtual que representa a transição entre dois pontos diferentes da MM (um ponto de terra e outro ponto de água). Para explicar o que acontece, são necessárias duas suposições: 1) a ferramenta está desenhando o mundo na posição  $(x, y, z)$ , que é o mapeamento do ponto  $(i, j)$  na MM, e 2) esse mapeamento corresponde a uma

faixa de água. À medida que a ferramenta incrementa a coordenada do mundo para poder desenhá-lo, supondo  $(x + 1, y, z)$ , esse ponto também é mapeado para a MM. Se o resultado do mapeamento da coordenada  $(x + 1, y, z)$  ainda for o ponto  $(i, j)$  na MM, então a ferramenta irá novamente desenhar um pixel de água na tela. Supondo que somente no ponto  $(x + 10, y, z)$  o mapeamento mude na MM para  $(i + 1, j)$  (e que esse ponto na MM seja terra), então todos os pontos anteriores a esse serão água e todos os pontos seguintes serão terra (até que o mapeamento na MM mude novamente).

A figura mostra claramente o momento em que o mapeamento na MM muda, que é quando a ferramenta substitui a renderização da água pela renderização da terra. Como não há algoritmos de geração de conteúdo atuando nessa transição, o usuário pode andar por essa linhareta da costa e não verá qualquer alteração em sua forma ou direção, a menos que outra quebra de mapeamento seja encontrada. Nesse caso, o usuário poderá ver outra costa retilínea perpendicular à aquela que ele está seguindo ou outra costa no mesmo sentido (que é a continuação da costa atual).

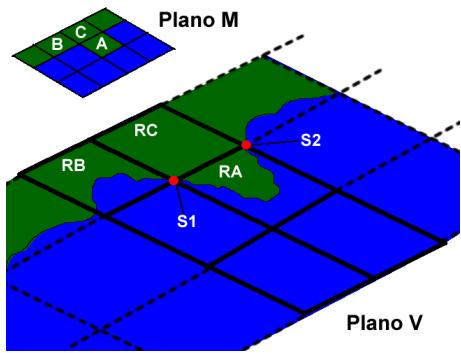
### 6.3.2 Quebra de linearidade da costa

O mapeamento dos pixels do mundo virtual com as informações da MM em baixa resolução resulta em efeito gráficos muito irrealistas. As praias naturais possuem uma curvatura característica e dificilmente terão um comprimento de 20Km em um configuração perfeitamente retilínea, como as praias geradas pela ferramenta. Embora o objetivo do presente trabalho não seja criar paisagens fotorrealistas, praias tão irrealistas não são aceitáveis. Para contornar esse problema, criou-se uma forma de quebrar a linearidade da costa através da adição de conteúdo aos locais onde o mapeamento da MM é calculado entre dois pontos, um de água e outro de terra. O funcionamento do algoritmo é descrito a seguir.

A MM possui a descrição completa do que é terra e do que é água no mundo virtual. Cada um de seus pixels possui um descritor associado, que informa aos demais algoritmos da ferramenta para qual tipo de terreno um pixel do mundo virtual está sendo mapeado na MM. Inicialmente a ferramenta foi criada com três tipos de terreno: água, terra (continente) e costa (terra em contato com a água). Depois que as faixas de terra são pré-processadas e essas informações são armazenadas na MM, existem apenas informações sobre terra (continente) e água.

A partir desse momento, o primeiro passo do algoritmo de quebra de linearidade da costa é executado. Utilizando como entrada a MM atual, o algoritmo varre cada um dos seus elementos, atualizando o descritor de informação dos pixels que são costa. Um pixel é dito costa quando pelo menos um de seus vizinhos é água. Depois que o algoritmo termina o seu processamento, a MM contém os três tipos de terreno descritos anteriormente. O próximo passo para a quebra de linearidade da costa é a geração de conteúdo com base no descritor de informação de cada um dos pixels da MM. Quando a ferramenta estiver gerando conteúdo para desenhar na tela, o procedimento de geração testa qual o tipo de terreno que está descrito no mapeamento da MM para os pixels que ela está desenhando atualmente. Se o mapeamento terminar em um pixel da MM que é terra, então a função irá gerar um relevo para aquele ponto. Se o mapeamento terminar em um pixel da MM que é água, então a função irá gerar o relevo para o oceano (que é uma altura padrão representando o nível do mar). Se o mapeamento terminar em um pixel da MM que é costa, então a função irá criar alterações nas informações de terra/água desse mapeamento, o que irá resultar em uma costa não-retilínea e mais realista. A figura 6 ilustra o funcionamento do algoritmo.

Os pixels A e B da MM possuem um descritor de informação indicando que eles são costa. Alguns outros pixels da figura também são costa, porém eles não são detalhados por fins de clareza. O Plano M descreve a MM e o Plano V descreve o resultado do mapeamento dela no mundo virtual. Embora não esteja descrito na ilustração, cada um dos blocos do Plano V é composto por vários pixels, enquanto cada bloco do Plano M corresponde a apenas um pixel da MM. O pixel C da MM é mapeado para um bloco maciço de terra no Plano V, já que o seu descritor informa à ferramenta



**Figure 6:** Funcionamento do algoritmo de quebra de linearidade da costa



**Figure 7:** Representação gráfica do espectro de ruído utilizado para criação da costa

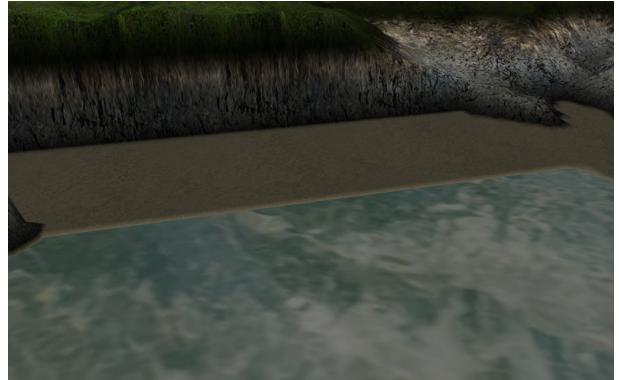
que ele é um pixel de terra. O pixel A seria mapeado para um bloco maciço de terra também, porém com a atuação do algoritmo de quebra de linearidade ele é mapeado para uma configuração diferenciada. No momento da geração de conteúdo para os pixels do mundo virtual que estão dentro do bloco RA, o algoritmo de quebra de linearidade distorce as informações de terra/mar para cada um dos pixels, o que faz com que o bloco não seja composto inteiramente de pixels de terra.

A implementação desse processo é baseada em ruídos e números aleatórios, na qual uma função paramétrica é usada para decidir o que é terra/mar dentro de um bloco que é mapeado para costa na MM. Utilizando como base a posição do pixel dentro do bloco RA, a função mapeia essa informação para dentro de um espectro de valores criados a partir de uma função de ruído de Perlin. Em linhas gerais, o que a função faz é testar se o hash do pixel em questão está dentro ou fora do espectro; o processo pode ser imaginado como um teste de altura em um *heightmap* de pequenas proporções (que é criado como resultado do espectro de ruído): se o retorno da função de ruído para o pixel em questão for maior que um determinado valor (que é a granularidade do bloco sendo analisado), então ele é terra, caso contrário ele é água. Quanto maior a granularidade do bloco, maior será a quantidade de terra no local. A figura 7 ilustra o desenho do *heightmap* fictício gerado pela função de quebra de linearidade no bloco RA.

### 6.3.3 Praias

A quebra de linearidade da costa elimina em grande parte o problema de linhas irrealistas nos continentes, porém o resultado final ainda tende para algo que não é aceitável na natureza. Quando a ferramenta está renderizando uma fatia do mundo, para cada pixels que é descrito como terra um relevo é associado a ele; o mesmo se aplica para os pixels que são descritos como água, porém nesse caso o relevo criado possui sempre a mesma altura (o nível do mar). Como uma consequência direta disso, se a ferramenta estiver desenhando um conjunto de pixels que descreve uma cadeia montanhosa e, logo em seguida, os próximos pixels são descritos como água, a paisagem resultante apresentará um "degrau". Isso acontece porque a cadeia montanhosa foi gerada muito próxima da água, o que faz com que a sua renderização seja abruptamente interrompida no momento que a ferramenta encontra água. Embora existam falésias no mundo real, elas não estão presentes em todas as costas, somente em algumas. Para contornar esse problema, criou-se um algoritmo capaz de gerar praias em terminadas áreas, o que torna a paisagem gerada mais realista.

O algoritmo de criação de praias atua pouco antes do conteúdo ser renderizado na tela. Depois que a ferramenta mapeia para a



**Figure 8:** Praia gerada ao longo da costa

MM os pixels e depois que o algoritmo de quebra de linearidade da costa atua, o resultado é um *heightmap* pronto para ser renderizado. Antes de ser desenhado na tela, esse *heightmap* é tratado pelo algoritmo de criação de praias. O procedimento varre cada um dos pixels existentes no mapa e, para cada um deles, checa qual a distância que o pixel atual está de um pixel de tipo água à sua volta, porém fazendo o mapeamento diretamente na MM, não no *heightmap*. A checagem é feita em quadro direções (direita, esquerda, cima e baixo), sendo que a ferramenta avança até encontrar um pixel água ou até que uma distância de N pixels seja percorrida. Em seguida, as quatro distâncias até um pixel de tipo água são somadas e utilizadas em uma fórmula para o cálculo da altura da praia. Os resultados possíveis são os seguintes:

- Se o pixel analisado estiver à uma distância de  $4N$ , isso quer dizer que a ferramenta percorreu às quatro direções possíveis e não encontrou água. Nesse caso, o pixel em questão não tem a sua altura recalculada; esse caso descreve o que acontece com todos os pixels que estão dentro do continente ou na costa porém longe da água: eles não formam uma praia e sua altura é definida pela função de relevo principal;
- Se o pixel analisado estiver à uma distância inferior a  $4N$ , então a sua altura será recalculada. Quanto maior for a distância calculada, maior será a altura do pixel, porém essa variação da altura é calculada dentro de um intervalo definido  $[T, B]$ , onde  $T$  é a altura máxima e  $B$  é a altura mínima de uma praia, respectivamente. O resultado dessa abordagem é uma praia que inicia alta no continente e, à medida que se aproxima da água, fica mais baixa.

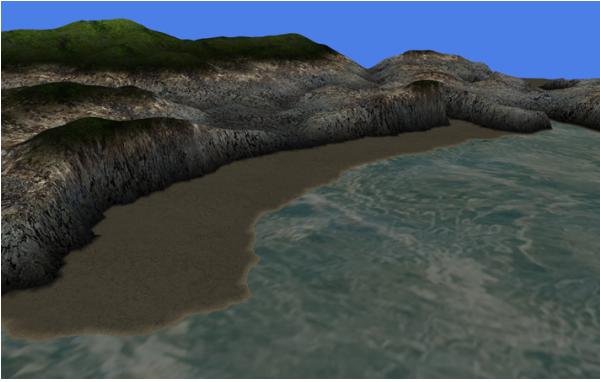
A figura 8 ilustra o resultado obtido com a geração de praias.

### 6.3.4 Arquipélagos e praias diferenciadas

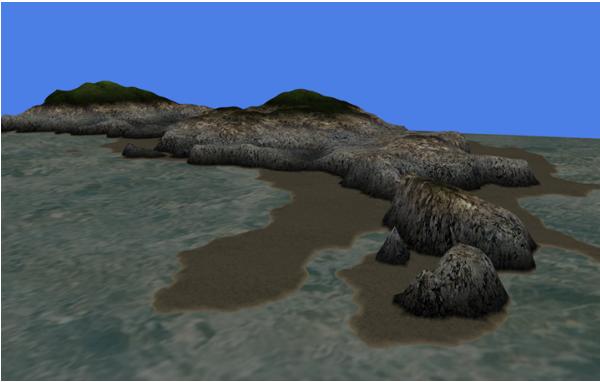
Utilizando os algoritmos de quebra de linearidade e criação de praias ao longo da costa, a ferramenta passou a gerar paisagens mais realistas. O resultado final em relação à costa e à praia, porém, apresentou um padrão muito definido, o que é incomum de acontecer no mundo real, no qual as linhas e paisagens naturais tendem a seguir um princípio aleatório ou menos padronizado. Se o usuário viajasse pelo mundo virtual apenas pela costa, ele veria praias com a mesma configuração (mesmo tamanho) e nenhuma ilha ou arquipélago ao longo do oceano. Para melhorar esse aspecto, criaram-se dois novos algoritmos que atuam na costa: um diferenciador de praias e um gerador de arquipélagos.

O **diferenciador de praias** atua perturbando a distância utilizada para o cálculo dos pixels água vizinhos a um determinado pixel. Em vez de utilizar uma distância fixa  $N$  para o cálculo da distância até a água, o diferenciador utiliza a posição do pixel como semente para uma função de ruído, que tem como retorno a nova distância que será utilizada nos cálculos. Utilizando essa técnica, o diferenciador é capaz de alterar o tamanho e forma da praia, o que faz com que determinadas regiões apresentem uma maior quantidade de areia do que outras. A figura 9 ilustra os resultados obtidos.

O **gerador de arquipélagos** atua criando novas faixas de terra em



**Figure 9:** Resultado obtido com a aplicação do diferenciador de praias



**Figure 10:** Ilha obtida com o gerador de ilhas

terminados pixels da MM. Depois que a MM é criada e todos os descriptores de informação de cada um de seus pixels é configurado, o gerador de arquipélago itera sobre os pixels que representam a costa e, para alguns deles, adiciona a informação que essa região possui ilhas. No momento em que a ferramenta estiver renderizando os pixels que são mapeados para essa região da MM, o descritor de informação será consultado e a ferramenta saberá que essa região necessita de um conteúdo novo, além do conteúdo utilizado para a quebra de linearidade da costa. Esse conteúdo é criado utilizando a mesma abordagem do algoritmo de quebra de linearidade, porém utilizando como pixels de análise aqueles pixels que são água. Para cada pixel sendo analisado, a sua posição é utilizada como um hash que é testado contra um espectro criado por uma função de ruído. Os ruídos utilizados para esse espectro são diferentes daqueles do algoritmo de quebra de linearidade, visto que o resultado esperado são pequenas porções de terra, não um bloco maciço dela. Para conseguir esse efeito, aumentou-se o número de oitavas da função de ruído de Perlin e aumentou-se o limite que é utilizado nos testes para saber se o pixel é ou não terra. O resultado do gerador de arquipélagos é combinado com o algoritmo de quebra de linearidade, o que faz com que as ilhas sejam geradas, em determinados casos, muito próximas à costa. O resultado final obtido com o gerador de arquipélagos são ilhas de diversos tamanhos ao longo da costa em determinados locais do mundo virtual. A figura 10 ilustra os resultados obtidos.

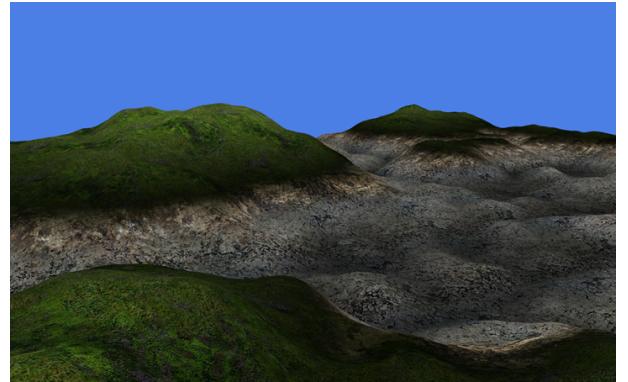
## 7 Resultados

Essa seção tem por objetivo avaliar cada uma das técnicas utilizadas na geração de conteúdo da ferramenta, mostrando os resultados que foram obtidos com cada abordagem. Leva-se em consideração a flexibilidade proporcionada pela ferramenta durante a utilização da funcionalidade e o resultado gráfico alcançado.

É importante frisar que o objetivo da ferramenta, no escopo do presente trabalho, não é a geração de conteúdos reais ou foto-realísticos, mas sim de elementos que possam ser utilizados para a criação de um cenário num jogo 3D. Entende-se por graficamente



**Figure 11:** Continentes e oceanos gerados pela ferramenta



**Figure 12:** Relevo gerado pela ferramenta

aceitável todo o resultado obtido que se enquadre dentro de um jogo e não destoe daquilo esperado pelo jogador, como uma montanha composta por ondulações senoidais suaves ao invés de um conjunto de cristas piramidais.

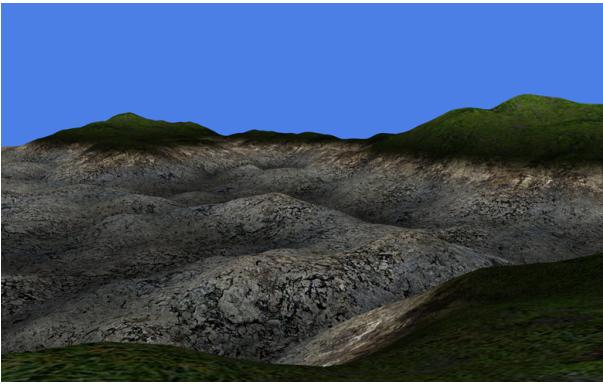
### 7.1 Avaliação dos continentes

O tempo para a geração dos continentes é diretamente proporcional ao tamanho da macro-matriz especificada. Em testes realizados, a redução da macro-matriz para valores inferiores a  $800 \times 800$  rendeu aumentos de desempenho consideráveis. Em contra partida, quanto menor o tamanho da macro-matriz, mais quadrados e lineares serão as linhas da costa de cada um dos continentes, o que pode produzir um resultado gráfico não muito aceitável. Para contornar esse problema, é possível ajustar o algoritmo de geração da costa para que ele faça alterações mais agressivas nas bordas dos continentes, o que irá quebrar a linearidade gerada por uma macro-matriz de baixa resolução. O ajuste desses dois elementos abre um leque de possibilidades para o programador, que pode encontrar um ponto de equilíbrio entre tempo de geração dos continentes e linearidade da costa. A figura 11 mostra os resultados obtidos em relação a continentes e oceanos gerados pela ferramenta.

### 7.2 Avaliação do relevo

O relevo gerado pela ferramenta é inteiramente parametrizável e customizável pelo programador. Segundo o protótipo da função esperada pela ferramenta para o cálculo do relevo, o programador pode criar qualquer código para a geração de relevo, o que garante flexibilidade nesse aspecto. A ferramenta possui embutida um gerador de relevo baseado na função de ruído de Perlin, que produz uma paisagem semelhante àquela encontrada no mundo real. A figura 12 ilustra o relevo criado pela ferramenta utilizando-se as funções de geração embutidas.

A função de relevo que foi implementada na ferramenta é capaz de gerar paisagem que imitam de forma aceitável o que é encontrado na natureza, porém a grande maioria dos esforços de desenvolvimento não foram empregadas sobre esse tópico. Como con-



**Figure 13:** Área com relevo de baixa frequência

sequência, a ferramenta não possui um grande leque de possibilidades de geração de conteúdos para o interior dos continentes, como cadeias montanhosas, planícies e planaltos. O relevo gerado atualmente é simplista no sentido de não apresentar grandes diversidades geológicas, entretanto se o objetivo da utilização da ferramenta não for criar um terreno rico em diversidade, o objetivo é plenamente atingido.

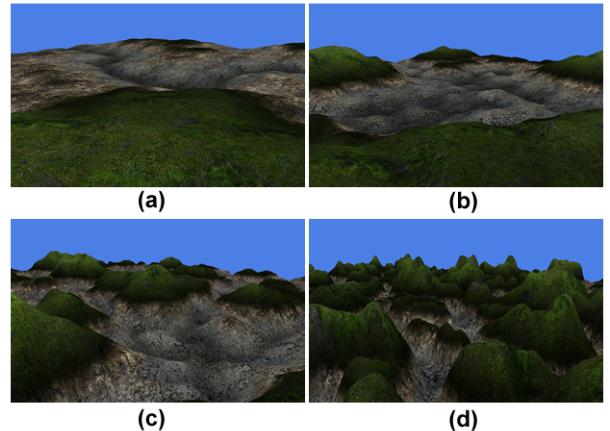
A geração atual de relevo foi desenvolvida visando-se a obtenção de ondulações de média/baixa frequência nas funções de ruído, o que faz com que não existam cadeias montanhosas pontudas ou depressões abruptas. A baixa frequência nos ruídos da função de geração de relevo faz com que o usuário veja montanhas com um ângulo muito suave de inclinação, além de áreas com poucas ondulações, que podem ser interpretadas como planaltos. A figura 13 ilustra uma área do mundo virtual que possui relevo com baixa frequência.

Se a função de geração de relevo utilizar frequências muito baixas e o seu espectro de valores for estendido à todo o mundo virtual, as montanhas e ondulações que serão produzidas serão muito suaves. Esse fenômeno acontece porque baixas frequências não criam alterações acentuadas de altura nos polígonos do relevo. Na utilização de altas frequências, a grande quantidade de ruído cria montanhas mais pontiagudas, porém as áreas intermediárias entre elas são muito ondulada, que é o reflexo da propagação de um espetro de valores muito ruidoso para o mundo virtual. Uma das formas de contornar esse problema é replicar a extensão do espetro de valores, ou seja, em vez de utilizar um espetro que cubra o mundo inteiro, utilizar esse mesmo espetro três vezes ao longo do mundo. Isso permite que pouco ruído seja utilizado, porém evita que o relevo resultante tenda ao plano. A ferramenta utiliza essa abordagem para criar um relevo com montanhas consideravelmente onduladas, porém sem áreas de ondulação estranha entre elas. A figura 14 ilustra os diferentes tipos de relevo obtidos com a variação do tamanho do espetro de valores.

### 7.3 Avaliação da costa

A geração da costa é composta por dois pilares principais, um de aspecto mais global e outro mais local. No aspecto global, a ferramenta utiliza apenas dados encontrados na MM para criar as linhas que compõem a costa, conforme descrito na seção 6.3.1. O resultado final para essa abordagem são costas completamente retilíneas, o que é irreal do ponto de vista do usuário. A figura 15 ilustra duas linhas da costa completamente retilíneas.

Depois que o método para quebra de linearidade da costa foi implantada, a ferramenta passou a apresentar paisagens mais aceitáveis. As figuras 16, 17 e 20 ilustram a criação de pequenas baías em certos locais da costa resultantes da aplicação do algoritmo. Isso acontece porque nesses locais o algoritmo de quebra de linearidade da costa criou braços de terra curvos partindo da linhareta do continente e, ao mesmo tempo, o algoritmo de criação de praias reduziu ao máximo a quantidade de areia encontrada no local. A ferramenta também é capaz de criar golfos, que são baías de grandes proporções, porém não é possível prever o local exato



**Figure 14:** Diversos relevos gerados a partir de diferentes espetros de ruídos. (a) Pouco ruído estendido para o mundo inteiro sem replicação. (b) Pouco ruído replicado 200 vezes. (c) Bastante ruído com replicação de 200 vezes. (d) Bastante ruído sem replicação.



**Figure 15:** Local de encontro de duas linhas da costa sem a aplicação de qualquer algoritmo de geração de conteúdo extra



**Figure 16:** Pequena baía com rochas



**Figure 17:** Baía de médio porte gerada a partir da criação de um braço de terra originada no continente

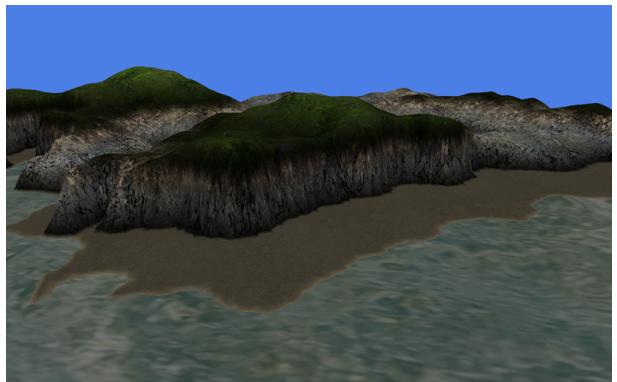
que isso irá acontecer, porque tal resultado depende da combinação de valores e coordenadas que variam sensivelmente ao longo do mundo virtual.

As figuras 18 e 19 ilustram a adição de conteúdo à costa retilínea do continente, resultando numa paisagem mais convincente para o usuário. As duas figuras mostram com bastante detalhe a combinação da função de geração de relevo, o diversificador de praias e o algoritmo de quebra de linearidade da costa na criação de conteúdo; a figura 18 apresenta uma rocha imediatamente à esquerda da falésia principal, resultado obtido a partir de um espectro de ruído que gerou dois pontos principais indicando a existência de terra: o maior sendo a falésia e o menor sendo a rocha grande à esquerda. Além disso, o diversificador de praias removeu a areia na parte de baixo da falésia, porém à esquerda ele fez o processo contrário. Em conjunto, o algoritmo de geração de relevo com replicação de valores criou pontas na extremidade esquerda da falésia.

A 19 apresenta a mesma combinação do algoritmo da figura anterior, porém o resultado obtido variou em função da coordenada do mundo virtual no local. Ao contrário do que aconteceu anteriormente, o diversificador de praias adicionou areia à base da falésia e, à esquerda da imagem, criou uma extensão da praia em forma de "braço". Também à esquerda, o diversificador de praias removeu a areia que fica na base do continente, criando um aspecto menos padronizado de conteúdo. À direita, é possível observar que o algoritmo de geração de relevo criou um declive que termina de forma relativamente suave na praia (relevo de baixa frequência), ao passo que na parte esquerda da imagem o relevo termina de uma forma mais abrupta (relevo de alta frequência).



**Figure 18:** Extremidade de um continente



**Figure 19:** Costa de um continente com praias de tamanho variável



**Figure 20:** Baía criada a partir da geração de dois braços de terra originados no continente

## 8 Conclusão

A criação automatizada de mundos virtuais procedimentais é uma das formas existentes para auxiliar desenvolvedores de jogos a criarem ambientes mais ricos em detalhes, em menos tempo e com menos recursos humanos. Ao contrário da abordagem puramente

não automatizada, no qual um *game designer* ou artista deve desenhar e modelar o mundo virtual por completo, na abordagem automatizada a adição de detalhes e modelagem de locais fica a cargo da ferramenta geradora. Dependendo do grau de realismo e da complexidade dos algoritmos de geração de conteúdo, uma ferramenta automatizada pode ser capaz de gerar um mundo virtual aceitável para diversos escopos de projeto.

Existem diversas pesquisas em relação à essa área, com abordagens diferentes e focadas em resultados diferenciados. Pode-se destacar nelas a preocupação em criar um mundo virtual de proporções infinitas sem que isso comprometa o desempenho da aplicação. Para atingir esses resultados, diversas técnicas são utilizadas, como aplicação de LOD e quadtrees para gerenciamento de malhas, além de diversos algoritmos procedimentais para a geração de conteúdo, como criação de relevo a partir de fractais, multi-fractais e/ou combinação de funções de ruído.

Este trabalho apresentou o desenvolvimento de uma ferramenta capaz de gerar mundos virtuais pseudo-infinitos com diversificação de formas e relevos ao longo de sua extensão. Utilizando uma combinação de algoritmos e métodos de gerenciamento de conteúdo, a ferramenta é capaz de criar praias, ilhas/arquipélagos, baías e costas que imitam as paisagens encontradas na natureza. Além disso, a possibilidade de parametrizar cada um desses elementos dá ao desenvolvedor um controle maior sobre o resultado que será obtido.

Dentre as inovações apresentadas, estão a criação de um terreno virtual de vastas proporções com enfoque mais detalhado no que diz respeito à geração da costa. Fruto de um aprofundamento nas pesquisas já realizadas na área, o desenvolvimento da ferramenta foi focado em tratar de forma diferenciada a criação de conteúdo para os diversos elementos existentes (como continentes, relevo, etc). Uma das prioridades do trabalho foi a criação de bordas dos continentes, não a criação de conteúdo para o seu interior; esse relevo está aceitável, porém ainda está muito aleatório e sem grandes resultados. A ferramenta é capaz de gerar continentes contendo falésias, praias e rochedos em suas extremidades. Em testes realizados, o tempo que um jogador levaria para chegar de uma ponta do mundo virtual até a outra num ambiente com o tamanho máximo suportado por um inteiro sem sinal, andando 100 pixel por segundo, seria de um 1 ano e 3 meses.

## References

- BLIZZARD ENTERTAINMENT, 2007. World of warcraft. Disponível em: <http://www.blizzard.com/>.
- CLARK, N. L. 2006. *Addiction and the Structural Characteristics of Massively Multiplayer Online Games*. Master's thesis, University of Hawaii, Hawai.
- COHEN, M., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. In *Siggraph 03 Conference proceedings*.
- DOLLINS, S. C. 2002. *Modeling for the Plausible Emulation of Large Worlds*. PhD thesis, Brown University, Estados Unidos da América.
- DUCHENEAUT, N., YEE, N., NICKELL, E., AND MOORE, R. J. 2006. Alone together exploring the social dynamics of massively multiplayer online games. In *CHI 2006 Proceedings*.
- GLASSNER, A. S., Ed. 1989. *An introduction to ray tracing*. Academic Press Ltd., London, UK, UK.
- GREUTER, S., PARKER, J., STEWART, N., AND LEACH, G., 2005. Realtime procedural generation of 'pseudo infinite' cities.
- GRIFFITHS, M. D., DAVIES, M. N., AND CHAPPELL, D., 2003. Breaking the stereotype the case of online gaming.
- HÄGGSTRÖM, H. 2006. *Real-time generation and rendering of realistic landscapes*. Master's thesis, University of Helsinki, Finlândia.
- HÄGGSTRÖM, H., 2009. Skycastle - free multiplayer game engine focusing on player creativity and world simulation. Disponível em: <http://www.skycastle.org/>.
- JASONWEBER, AND PENN, J., 1995. Creation and rendering of realistic trees.
- LEFEBVRE, S., AND NEYRET, F. 2003. Pattern based procedural textures. In *Proceedings of the 2003 symposium on Interactive 3D graphics*.
- LINDA, O. 2007. Generation of planetary models by means of fractal algorithms. Tech. rep., Czech Technical University.
- LINTERMANN, B., AND DEUSSEN, O., 1998. A modelling method and user interface for creating plants. Computer Graphics Forum.
- MOGENSEN, T. ., 2009. Instant planet generator. Disponível em: <http://www.eldritch.org/erskin/roleplaying/planet.php>.
- MUSGRAVE, F. K., KOLB, C. E., AND MACE, R. S., 1989. The synthesis and rendering of eroded fractal terrains. Computer Graphics, Volume 23, Number 3, July 1989, pages 41 to 50.
- OLSEN, J., 2004. Realtime synthesis of eroded fractal terrain for use in computer games.
- OPENGL, 2007. Opengl shading language. Disponível em: <http://www.opengl.org/documentation/glsl/>.
- PRZEMYSŁAW, P., AND LINDEMAYER, A., 1990. The algorithmic beauty of plants. Springer-Verlag.
- SONY ENTERTAINMENT, 2007. Everquest. Disponível em: <http://everquest2.station.sony.com/>.
- WANG, T., 2000. Integer hash function. Available at: <http://www.concentric.net/Ttwang/tech/inthash.htm>.