Charack: tool for real-time generation of pseudo-infinite virtual worlds for 3D games

Paper 60326

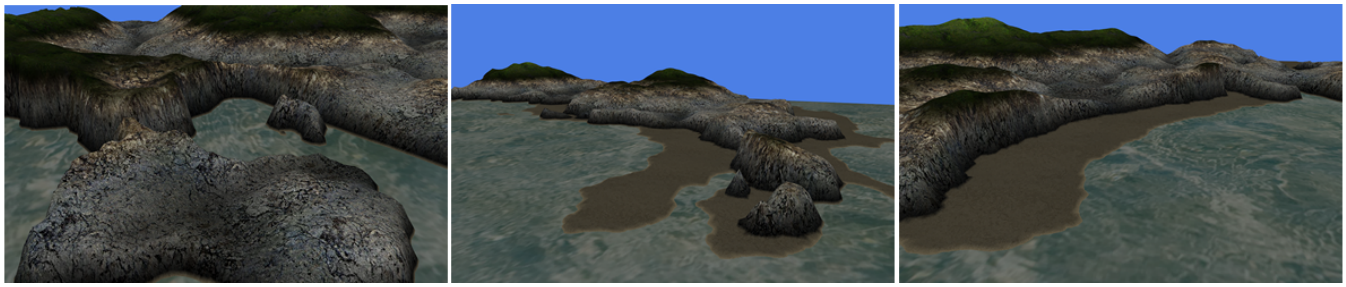

Figure 1: *Coastlines and islands procedurally generated by Charack*

Abstract

In MMO games the player's experience is mainly influencied by the size and details of the virtual world. Technically the bigger the world is, the bigger is the time the player takes to explore all the places. This work presents a tool able to generate pseudo-infinite virtual worlds with different types of terrain. Using a combination of algorithms and content management methods, the tool is able to create beachs, islands, bays and cost lines that imitates the real world landscapes. One of the contributions of the tool is the ability to generate giant pieces of land focusing on the coast line generation. The development of the present work aimed to handle separately the generation of content for all elements in the world (continents, terrains, etc.). The main point in the work is the coast line generation, not the content inside the continents.

Keywords:: MMO, virtual worlds, terrain generation, 3D games, noise, procedural generation, multifractal

Author's Contact:

1 Introduction

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.

2 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 an 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.

3 Related work

3.1 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. Whe 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.

3.2 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 to 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 approacha a texture cell is created so that the a 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 consequene, border textures will overlap the some reagions 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].

3.3 Procedural planets

The use of fractals and their combinations with other methods are also used to proceduraly 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 featuringits 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).

3.4 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.

3.5 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.

3.6 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.

4 Tool organization

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

4.1 Initial analysis

There are several related works concerning the generation of virtual worlds, whether finite or infinite. One of the related work 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 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 heightmap 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 heightmap 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 heightmap generation, so we believe the Charack contribution will be better if the efforts are focused on the generation of continents and coastlines.

4.2 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 2 shows Charack basic structure.

4.2.1 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 3.6. 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

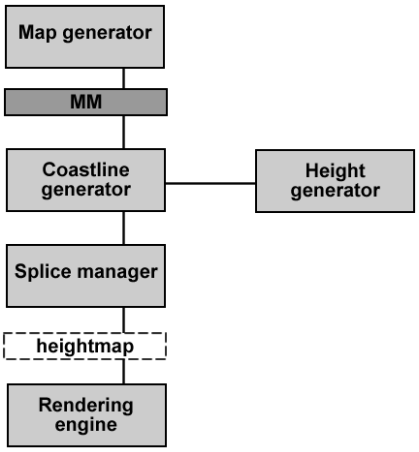

Figure 2: *Charack's basic structure*

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

4.2.2 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.

4.2.3 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.

4.2.4 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 and returns it 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 sevel pixels in the virtual world. Figure 3 illustrates the MM mapping process.

As the figure ilustrates 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 `1000x1000` as its size and the MM had `10x10` 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 `100x100` 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

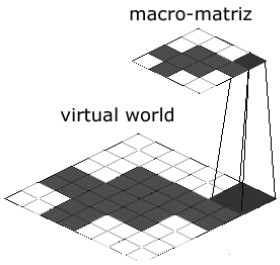

Figure 3: *Mapping the MM to the virtual world: each MM pixel represents several pixels in the virtual world.*

return it with different values, which will result in a coast line for that area, not am area entirely filled with land or water.

4.2.5 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.

5 Implementation

The main problem concerning Charack's implementation was the on demand content generation. Based on the fact the user can only see what is inside the view area, all the content generation algorithms need to take into account *only* the information that is available in the user's view. Even though this approach is efficient for resources management (process only the visible elements), it increases the complexity of the content generation algorithms.

The algorithm that generates mountains, for instance, has no way to determine where the mountain ends, because the world outside the user's view technically does not exist yet, it will be generated as the user moves. One approach to solve that problem would be the use of a function that describes the mountain chain, but this function should not rely on begin/end points, because they could not exist in a certain time. If that function does not need any begin/end points, at least it would have to rely on the position of the user in the virtual world. If the function must be aware of some special points, those points have to be previously processed, which would break the on demand content generation concept.

In addition the algorithms are drastically affected by the fact that the information they receive in a certain time may disappear altogether in the next iteration, since the user can move and change the visible content. Using the example of the mountain generation, a mountain could present an abrupt end, because the points being used for the content generation left the user's view.

To circumvent these problems the content generation was divided into three main stages: infinite terrain, continents and height generation. The on demand content generation affects differently each of these stages and the problems and solutions related to each stage are described in the following sections.

5.1 Infinite terrain

The main idea for the content generation is allow the user to walk in a infinite way, looking at new content each time a significant movement is performed. As the user walks, the tool must be able to identify where the observer is located in the world in order to generate the content around that position. To solve this problem, a variation of a technique described by [Greuter et al. 2005] was used.

The solution makes the player able to look at the screen and see a slice of the virtual world, but with no explicit divisions in the world, such as cells. Unlike what was done in [Greuter et al. 2005] where the user's view is a cone, the Charack user's view is a square centered on the user. Using the user position `(x, y, z)`, the visible content arround that coordinate is sliced from the world and

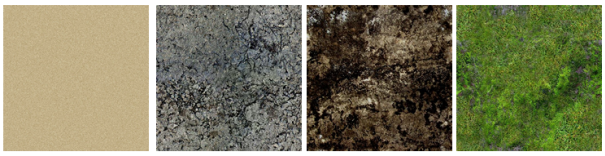

Figure 4: *Set of image used for terrain texturization*

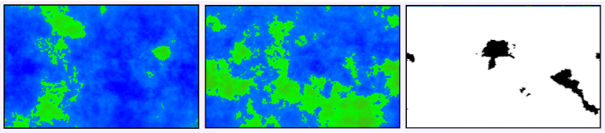

Figure 5: *Random planets generated with different seeds. (a) and (b) maps featuring height value information; (c) map featuring only information about what is wahter/ocean.*

drawn in screen. If the player reaches a place or a complete slice is not possible, such as the world boundaries, no content is displayed beyond the world area.

In order to texture the sliced data a set of images are interpolated and managed by the shading language GLSL [OpenGL 2007]. The height value of the pixel defines the interpolation weight of each texture. As a consequence, a sand texture has highter weight for pixels featuring a low height value, for instance. Figure 4 shows all the available textures.

5.2 Height generation

The main idea for the terrain height generation is the use a parametric function that informs the height value of each vertex. The function is seeded with the point location in the world. As a consequence, the function is able to describe all the height information in the world with no limitions concerning the world size. The processing time is related to the size of the user's view, not related to the world size, since the function uses the point information to calculate its height value. The height values are generated with a Perlin noise function [Perlin 1985].

5.3 Continents

The generation of continents and oceans has been proposed in order to break the monotony of a landscape composed only by land and to increase the similarity of the virtual world with the real world landscapes. The solution for the continents generation consist in pre-process the land areas and store that information for other tool calculations. With that approach, the on demand content generation has been partially broken, since the continents are generated before all the other content, but it ensures a better control over water/land areas.

The continents generation is based on the approach described in section 3.6. That planet generator was used because it has several parameterization options, such as the possibility to use a seed to manage all the random calculations. Figure 5 *(a)*, *(b)* and *(c)* illustrates the results obtained with the planet generator of [Mogensen 2009].

5.3.1 Problems with continent generation

As previously explained in section 4.2, each pixel of the MM is mapped to several pixels in the virtual world. A direct consequence of that mapping process is the generation of large areas featuring straight land lines. If there were no hardware limitation and if it were possible the generation a matrix with the exact size of the virtual world, the matrix would contain the necessary resolution for the tool to accurately determine whether a pixel is land or not land, in a ratio of 1:1 (one MM pixel is mapped to one world pixel). This approach, however, is not suitable because a matrix with such proportions consumes many resources and processing time. Although

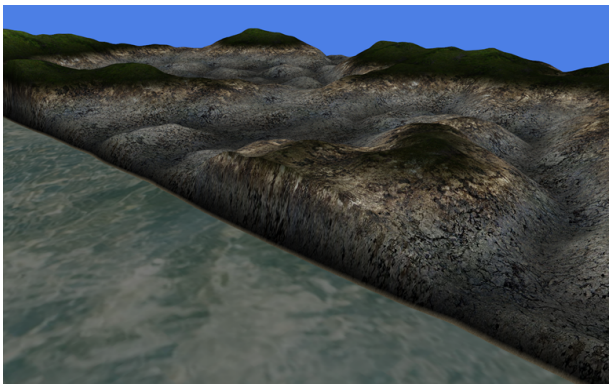

Figure 6: *The result of no algorithm to generate extra content to fill the discrepancies in the MM mapping process*

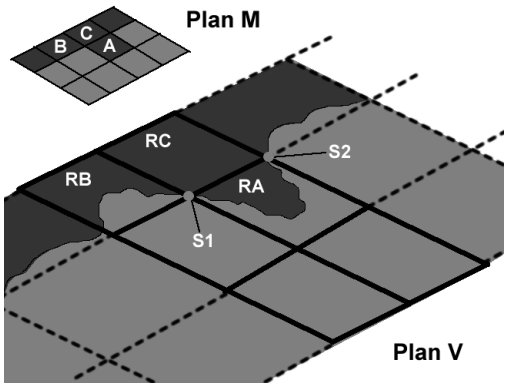

Figure 7: *Coastline disturbance algorithm*

Figure 8: *Small heightmap generated by the costline disturbance algorithm*

the tool allows customization of the MM size, tests presented that a `800x800` pixels MM has is quite enough information to be processed by all the other algorithms of the tool.

Figure 6 illustrates the results obtained by the tool when no algorithm is used to generate extra content to fill the rectilinear spaces of the virtual world.

This figure ilustrates a place in the virtual world that represents the transition between two different points of the MM (a land point and a water point). To explain what is happening, assume the tool is drawing the world at position `(x, y, z)`, which is the mapping result of a pixel `(i, j)` in the MM, which is described as land; as the tool increases the coordinate in order to draw the landscape, each new position is mapped to the MM. If the result of the mapping process of new coordinate, `(x + 1, y, z))` for instance, is still the point `(i, j)` in the MM, then the tool will again draw a land pixel on the screen. Assuming that only at point `(x + 10, y, z)` the pixels start being mapped to a different pixel in the MM, such as `(i + 1, j)` (and `(i + 1, j)` is described as land), then all points before the position `(x + 10, y, z)` are drawn as water and all pixels after that location are drawn as land.

The figure shows clearly when the world coordinates start beeing mapped to a different entry in the MM, which is when the tool replaces the lang rendering with water rendering. As a consequence of no algorithm being applied to generate content for that transition area, the user will move along the coastline and will see only straight lines.

5.3.2 Coastline disturbance

The mapping process of the pixels of the virtual world to the MM produces very unrealistic landscapes. A real world beach has a natural curvature and hardly have a length of 20km in a perfectly straight configuration, as the beaches generated by Charack. Although the objective of this work is not to create photo-realistic landscapes, such unreal beaches are not acceptable. To circumvent this problem, a coastline disturbance is applied to the locations where the mapping process is made between two MM points, one of them described as land and the other one described as water. The algorithm is described below.

The MM has a full description of what is land and what is water in the virtual world. Each of its pixels has a descriptor, which tells the other algorithms what type of terrain one pixel of the virtual world is after it is mapped to the MM. Charack features three types of terrain: water, land (continent) and offshore (land in contact with water). After the continents are pre-processed and stored in the MM, it only features information about land (continents) and water.

From that moment, the first step of the coastline disturbance algorithm is performed. Using the current MM as its input, the algorithm scan each MM's pixel, updating the descriptor of pixels that represent a coast. A pixel is said to be coast when at least one of its neighbors is water. After the algorithm end, the MM contains the three types of terrain described before (water, land and offshore).

The next step to apply disturbance to the coastline is the content generation based on the descriptor of each pixel in the MM. When the tool is creating content to draw on the screen, each pixel being drawn is tested against its descriptir in the MM. If the pixel is mapped to a land pixel in the MM, then the function will set a height value for that point. If the pixel is mapped to a water pixel in the MM, then the function will set the sea level height to that point. Finally if the pixel is mapped to a offshore pixel in the MM, then the function will disturb the land/water information of that mapping process, which will result on a non-straight coast line. Figure 7 illustrates the algorithm.

The MM pixels `A` and `B` have a descriptor indicating that they are described as a coast. Some other pixels of the figure are also coast, but they will not be detailed for understanding purposes. Plan `M` describes the MM and plan `V` describes the result of the mapping process between them. It is not described in the figure, however each blocks of plan `V` is composed of several pixels, while each block of plan `M` represents only one MM pixel. The MM pixel `C` is mapped to a massive block of land in the plan `V`, as its descriptor tells the tool that it is a pixel described as land. The pixel `A` would also be mapped to a massive block of land, but with the intervention of the coastline disturbance algorithm it is mapped to a different configuration. During the content generation for the pixels that are inside the block `RA`, the coastline disturbance algorithm alters the land/sea information for each pixel, so that the block will not be composed of land or water pixels only, but a combination of them instead.

The implementation of that process is based on a noise function and random numbers with a parametric function deciding what is land and what is water for all pixels described as a coast in the MM. Using the pixel position in the block `RA`, the function maps that information into a spectrum of values created by a Perlin noise function. What the parametric function does is check if the hash of the pixel being analyzed is inside or outside of the spectrum. The process can be imagined as a height test against a small heightmap (which is created as result of the noise spectrum): if the return of the noise function for that is greater than a certain value (which is the granularity of the block being analyzed), then it is classified as land, otherwise it is classified as water. The higher is the granularity of the block, the greater is the amount of land on that location. Figure 8 illustrates the small heightmap generated by the costline disturbance algorithm when block `RA` is being processed.

5.3.3 Beaches

The coastline disturbance algorithm minimizes the problem of unrealistic continents lines, but the outcome is not quite good enought.

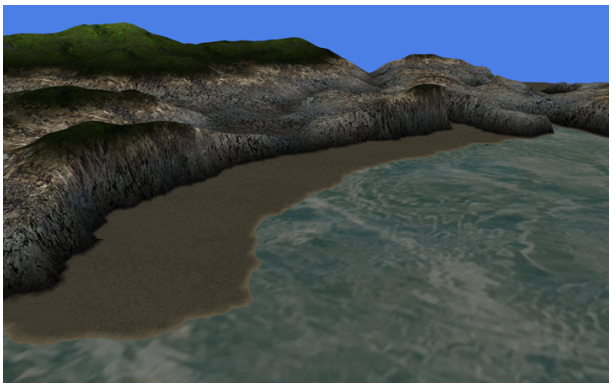

Figure 9: *Results of the beach generator algorithm*

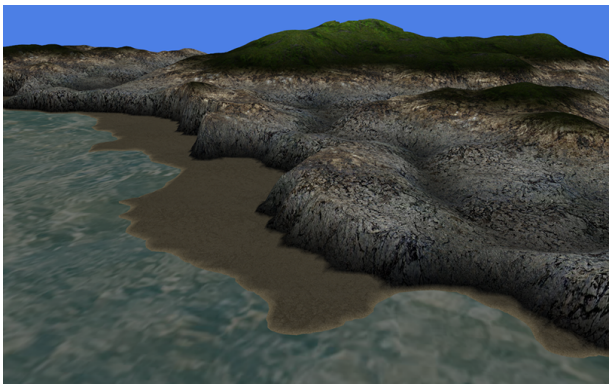

Figure 10: *Results of the beach disturber algorithm*

When Charack is rendering a slice of the world, for each pixel described as a land a height vaule is set to the pixel; the same applies to the pixels that are described as water, but in that case the height value is always the same (the sea level). As a direct result of that approach if the tool is drawing a set of pixels describing a mountan and the next pixels are described as water, the landscape will features a ”step”. It happens because the mountain chain was generated very close to the water, which means that its rendering is abruptly interrupted when Charack finds pixels described as water. Although there are cliffs in the real world, they are not present in all coasts. To solve this problem, a special algorithm is applied in order to create beachs in certain locations of the world, which makes the generated landscape looks more realistic.

The beach generation algorithm is performed right before the content is rendered on the screen. After Charack maps the pixels to the MM and after the coastline disturbance algorithm is performed, the result is a heightmap ready to be rendered. The heightmap is treated by the beach creator algorithm before being drawn on the screen. The procedure scans all the pixels in the map and for each one it checks the distance from the current pixel to a near water pixel. The pixels around the target are mapped directly to the MM, so the only information that is used from the heightmap is the pixel location in the world (which is necessary to map it to the MM). The checking process is performed in four directions (right, left, up and down) and it ends when a water pixel is found or when `N` pixels were analyzed. After that, the four distances are added and used to calculate the height of the beach. The possible results are:

- If the pixel has a value of `4N` for its distance, it means the tool has iterated through to the four possible directions and found no water. In this case, the height value for the pixel remains the same. It happens to all the pixels that are within the continent or on the coast but away from the water: they do not belong to the beach area and their height value is defined by the height generator;
- If the pixel has a value smaller than `4N` for its distance, then its height value will be recalculated. The greater the distance from that pixel to the water pixel, the greater is the height value that will be applied. The height variation is calculated within a range of `[T;B]`, where `T` is the maximum height and `B` is a minimum height value of all pixels in the beach. The result of that approach is a beach featuring higher height values near the continent and lower height values near the water.

Figure 9 shows the results of the beach generator algorithm.

5.3.4 Island generator and beach disturber

The coastline disturbance and the beach generation algorithms make Charack able to generate more realistic landscapes. The final, however, presents a well definite pattern, which is unusual to happen in the real world, where the lines and landscapes are more likely to follow random patterns. If the player walks in the virtual world only through the coast, he would see beaches with the same configuration and no islands along the path. To avoid that problem, two new algorithms are applied to the coastline pixels: beach disturber and island generator.

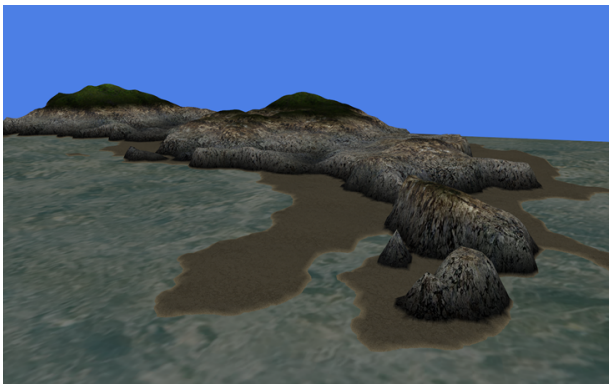

Figure 11: *Island generated by the island generator*

The **beach disturber** disturbs the distance used to calculate the water pixels neighboring a certain pixel. Instead of using `N` as value to calculate the distance from the pixel to the water, the beach disturber uses the pixel position as a seed and generates a new value that will be used as the distance. Using this technique, the beach disturber is able to change the size and shape of the beach, so that certain regions have a greater amount of sand than others may have. Figure 10 shows beach disturber results.

The **island generator** creates land portions in some MM pixels. After the MM is created and all the pixel descriptors are configured, the generator iterates through all pixels described as coast and for some of them it adds sets a flag describing that region as a place that features islands. Each pixel mapped to that special regions of MM has its position used as a hash that is tested against a spectrum created by a noise function. The noise spectrum used for that are different from that one used in the coastline disturbance algorithm, since the expected outcome are small portions of land (islands). The final result obtained with the island generator is a wide range of beach sizes. Figure 11 shows the island generator results.

6 Results

This section aims to evaluate each of the techniques used in the content generation process, explaining the obtained results for each approach. It is important to highlight that Charack's purpose is not the generation of real or photo-realistic content, but elements that can be used to create a 3D game scene. A result is classified as graphically acceptable if it can be integrated into a game and not surprive player in a negative way, such as a pyramidal mountain instead of a smooth mountain.

6.1 Continents evaluation

The time spent for the continent generation is directly proportional to the size of the specified MM. The reduction of the MM size to `800x800` yielded significant performance improvements. As

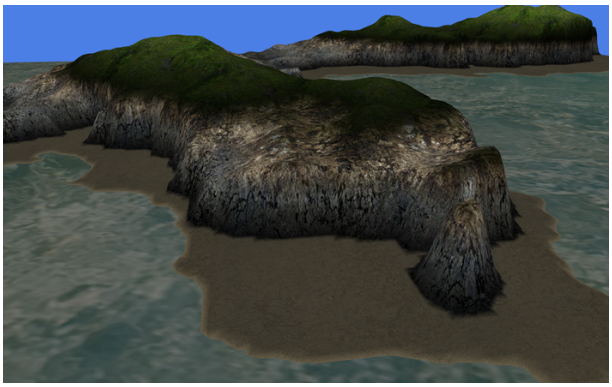

Figure 12: *Continents and oceans generated by Charack*

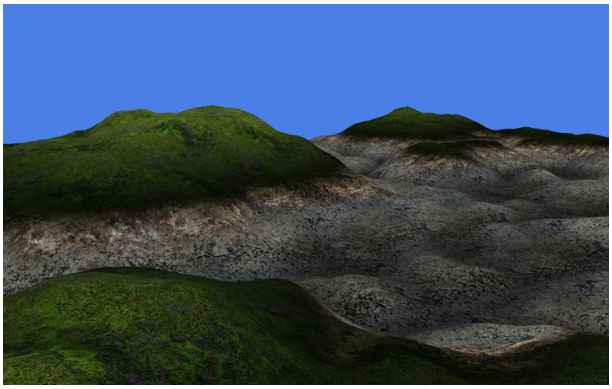

Figure 13: *Terrain heightmap created by Charack's built-in generator*

a consequence, the smaller is the MM size, the more linear and square are the coast lines of each continent. To avoid that problem, it is possible to adjust the coast line disturbance algorithm to make it produce more aggressive changes in the coastlines. Figure 12 shows the continents and oceans generated by the Charack.

6.2 Terrain height evaluation

The terrain height generated by Charack is fully customizable. The tool has a built-in terrain height generator based on Perlin noise designed for testing purposes only. The main focus of the present work is the continents and coastline generation, so any activity related to terrain height generation was very superficial and presents no contribution. Figure 13 shows the terrain height created by the built-in generator.

6.3 Coastline evaluation

The coastline generation is composed of two main elements, a global and a local one. The global one only uses only data available in the MM in order to create the coastlines, as described in section 5.3.1. The final result for that approach is aunreal straight coastline. Figure 14 shows two completely straight coastlines.

After the coastline disturbance algorithm was established, Charack started to produce more acceptable landscapes. Figures 15, 16 and 19 show small bays in some places of the coast. It happens because at those locations the coastline disturbance algorithm created pieces of land towards the ocean and at the same time the beach disturber reduced the amount of sand found on newly created land pieces. Charack is also able to create gulfs, which are large bays, but it is not possible to predict the exact location where those bays will happen, because it depends on a set of specific values (location, beach size, etc).

Figures 17 and 18 show final result obtained with the combination of all the previously described algorithms: coastline dirturbance, beach disturber and island generator.

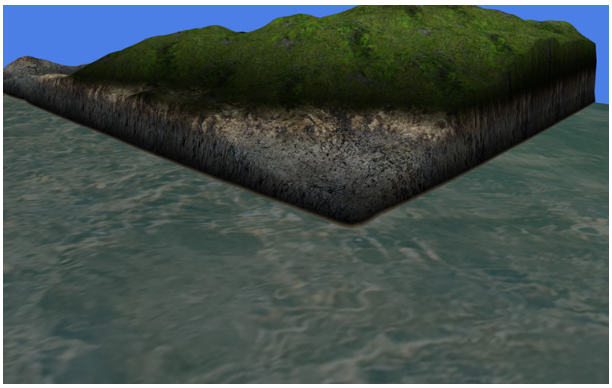

Figure 14: *The interception of two coastlines with no extra content being applied to them*

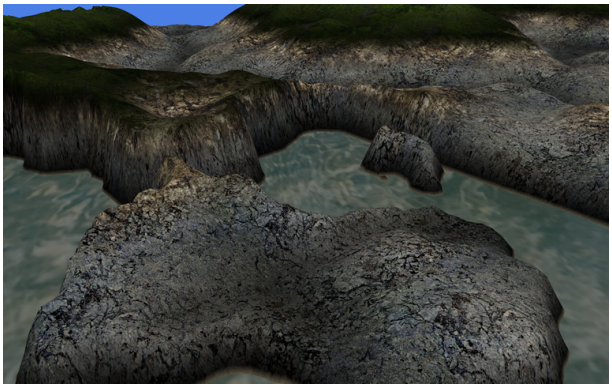

Figure 15: *Small bay featuring rocks*

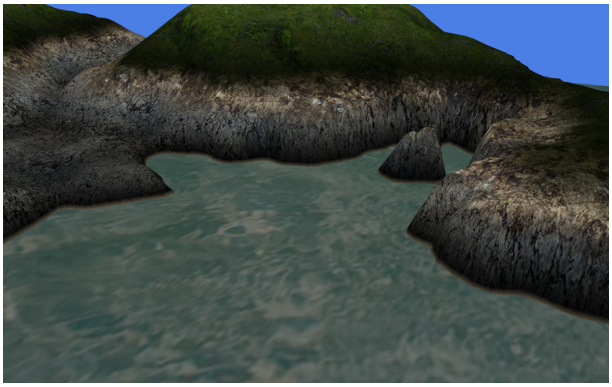

Figure 16: *Small bay featuring no beach area*

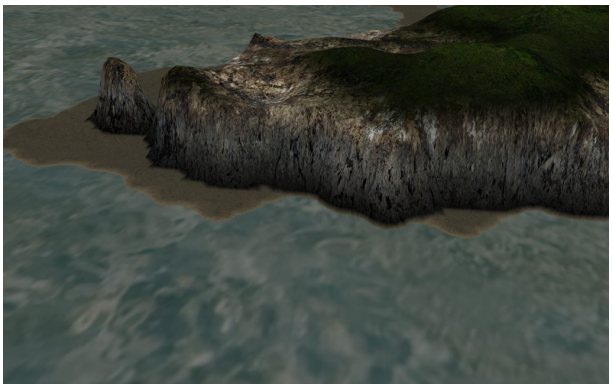

Figure 17: *Coastline featuring almost no beach area*

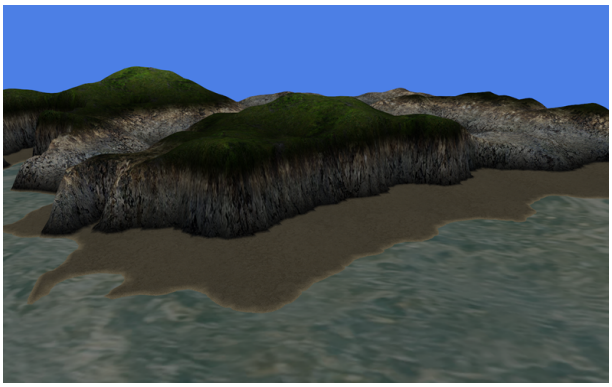

Figure 18: *Coastline featuring beachs with different sizes*

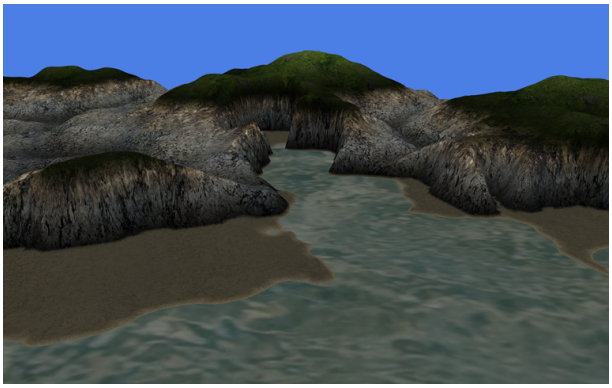

Figure 19: *Result of the coastline disturbance algorithm*

7 Conclusion

The automated creation of virtual worlds is one of available methods that can help developers to create games featuring detailed environments in less time and using fewer resources. Unlike the purely non-automated approach where a game designer has to to draw the entirely virtual world, an automated approach is able to generate a complete world with none of almost none human interference. There are several researches on that area using different approaches and focused on a wide range of results. It is possible to highlight the concern to create an infinite virtual without compromising the performance of the application.

This paper presented a tool able to generate pseudo-virtual worlds featuring different continents, coastlines and landscapes. Using a combination of algorithms and methods for content management, the tool is able to create beaches, islands, bays and coastlines similar to the ones found in the real world.

One of the Charack's contributions is the ability to generate giant pieces of land focusing on the coastline generation. The development of the present work aimed to handle separately the generation of content for all elements in the world (continents, terrains, etc.). The main point in the work is the coastline generation, not the content inside the continents. A player featuring a 100 pixels per second speed in a virtual world generated with the maximum value allowed by a integer will take about 1 year and 3 months to across the whole world.

References

BLIZZARD ENTERTAINMENT, 2007. World of warcraft. Available at: http://www.blizzard.com.

CLARK, N. L. 2006. *Addiction and the Structural Characteristics of Massively Multiplayer Online Games*. Master's thesis, University of Hawai, 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. Available at: 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. Available at: 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. Available at: http://www.opengl.org/documentation/glsl/.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH*, 287–296.

PRZEMYSLAW, P., AND LINDENMAYER, A., 1990. The algorithmic beauty of plants. Springer-Verlag.

SONY ENTERTAINMENT, 2007. Everquest. Available at: http://everquest2.station.sony.com/.

WANG, T., 2000. Integer hash function. Available at: http://www.concentric.net/ Ttwang/tech/inthash.htm.