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

GERAÇÃO PROCEDURAL DE CENÁRIOS 3D DE CÂNIONS COM FOCO EM JOGOS DIGITAIS

DISSERTAÇÃO DE MESTRADO

Daniel Michelon De Carli

Santa Maria, RS, Brasil

2012

GERAÇÃO PROCEDURAL DE CENÁRIOS 3D DE CÂNIONS COM FOCO EM JOGOS DIGITAIS

Daniel Michelon De Carli

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação Aplicada, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Computação

Orientador: Prof. Dr. Cesar Tadeu Pozzer

Santa Maria, RS, Brasil

2012

Ficha catalográfica elaborada através do Programa de Geração Automática
da Biblioteca Central da UFSM, com os dados fornecidos pelo(a) autor(a).

Michelon De Carli, Daniel
GERAÇÃO PROCEDURAL DE CENÁRIOS 3D DE CÂNIONS COM FOCO EM JOGOS DIGITAIS / Daniel Michelon De Carli.-2012.
90 f.; 30cm

Orientador: Cesar Tadeu Pozzer
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2012

1. Geração Procedural 2. Terreno 3. Cenário 3D 4. Mean Shift 5. Algoritmo de Dijkstra I. Pozzer, Cesar Tadeu II. Título.

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

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

GERAÇÃO PROCEDURAL DE CENÁRIOS 3D DE CÂNIONS COM FOCO EM JOGOS DIGITAIS

elaborada por
Daniel Michelon De Carli

como requisito parcial para obtenção do grau de
Mestre em Computação

COMISSÃO EXAMINADORA:

Cesar Tadeu Pozzer, Dr.
(Presidente/Orientador)

Manuel Menezes de Oliveira Neto, Dr. (UFRGS)

Juliana Kaizer Vizzotto, Dra. (UFSM)

Santa Maria, 5 de Março de 2012.

DEDICATÓRIA

À minha esposa, Graziele C. Kemmerich

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer a Deus por me inspirar boas ideias e fazer com que eu conhecesse as pessoas certas que me ajudaram a trilhar esse caminho.

Gostaria de agradecer aos meus pais que, com muito esforço, dedicação e visão de futuro, me possibilitaram chegar até aqui.

Agradeço à minha esposa, Graziele C. Kemmerich, pelo auxílio, dedicação e companheirismo durante todos os momentos da minha vida, e em especial, deste trabalho. Palavras não são suficientes para expressar toda a gratidão que sinto por ela.

Agradeço imensamente meu orientador, Prof. Cesar Tadeu Pozzer, pela incansável tarefa de me orientar, de fazer com que eu buscasse neste trabalho sempre o meu melhor. Além de orientador é, acima de tudo, um exemplo para mim. Também agradeço ao Prof. Marcos d'Ornellas, pelo acolhimento e ajuda, que foram muito importantes no início do mestrado. Além deles, agradeço a Prof^a. Lisandra Manzoni Fontoura, pela ajuda e ensinamentos, que também foram muito importantes para o meu primeiro ano de mestrado.

Agradeço também ao meu amigo, Fernando Bevilacqua, pelas incontáveis trocas de ideias e revisões do texto. Com certeza, esta conquista não seria igual sem sua ajuda. Além dele, meu obrigado também ao Victor Schetinger, Diego Schmaedech, Guilherme Schardong e Bernardo Henz – colegas do LaCA – que sempre me ajudaram de alguma forma.

Por fim, agradeço à Universidade Federal do Pampa (UNIPAMPA) por me permitir realizar as atividades de analista juntamente com a vida acadêmica.

“Do or do not. There is no try.”
— MESTRE YODA

RESUMO

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

GERAÇÃO PROCEDURAL DE CENÁRIOS 3D DE CÂNIONS COM FOCO EM JOGOS DIGITAIS

AUTOR: DANIEL MICHELON DE CARLI
ORIENTADOR: CESAR TADEU POZZER
Local da Defesa e Data: Santa Maria, 5 de Março de 2012.

Esta dissertação propõe um método procedural não assistido, baseado em técnicas de computação gráfica, visão computacional e busca em grafos, para a geração de cenários 3D de cânions com foco em jogos digitais. Para definir as características a serem reproduzidas, foram analisadas diversas imagens de cânions reais chegando-se em dois modelos, um comum e outro recursivo. A abordagem proposta manipula um reticulado gerado com ruído de Perlin, moldando assim as características inerentes a essa formação geológica. São levadas em conta as diversas parametrizações necessárias para permitir que o algoritmo construa cânions com curso de rio, áreas de planícies, regiões de encosta suave, estruturas de penhascos e, por fim, planaltos nas regiões mais altas. Para atingir o resultado final, o trabalho utiliza o algoritmo *Mean Shift* como mecanismo de segmentação, definindo dados e regiões de interesse. Munido dos dados do algoritmo de clusterizacao, é definido um limiar para a criação de uma máscara binária com a definição das planícies. Em um segundo momento, um algoritmo de rotulação de componentes conectados é executado, extraindo-se os centróides de cada planície. Por sua vez, o algoritmo de Dijkstra encaixa-se na definição de rotas que conectam estas planícies. O algoritmo de Dijkstra é, então, executado novamente, tendo por base uma função de custo de inclinação, para definir o curso do rio. Por fim, uma filtragem espacial baseada em um filtro Gaussiano é aplicada para interpolar as regiões de encostas de declive suave. A combinação dessas técnicas gera terrenos com grande variabilidade e com as características inerentes à formação geológica de cânions.

Palavras-chave: Geração Procedural, Terreno, Cânion, Cenário 3D, Mean Shift, Algoritmo de Dijkstra, Ruído de Perlin.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

PROCEDURAL GENERATION OF 3D SCENES FEATURING CANYONS FOCUSED ON DIGITAL GAMES

AUTHOR: DANIEL MICHELON DE CARLI
ADVISOR: CESAR TADEU POZZER
Defense Place and Date: Santa Maria, March 5^{th}, 2012.

This Master's thesis proposes a non-assisted procedural method for 3D canyons' scenes generation based on techniques of computer graphics, computer vision and graph search algorithm. In order to define all the features to be reproduced in our scenes, we have analyzed several images of real canyons and have categorized them in two canyon features models: a recursive and an ordinary one. The proposed approach manipulates a heightmap, created using Perlin noise, in order to imitate the geological features formation previously analyzed. Several parametrizations are used to guide and constraint the generation of terrains, canyons features, course of river, plain areas, soft slope regions, cliffs and plateaus. This work also uses the Mean Shift algorithm as mechanism of segmentation to define regions of interest. A binary mask, with plain areas, is defined based on a threshold operation by a given data set provided by the Mean Shift algorithm. Thereafter a connected-component labeling algorithm is executed using the previously binary mask. This algorithm finds all plains' centroids. Right after that, the Dijkstra's algorithm is performed in order to connect all plain areas, creating a valid path between the centroids. The Dijkstra's algorithm is executed again to define the river's course. Finally, a Gaussian smoothing operation is applied to interpolate the soft slope regions. The combination of all those techniques produces as a result automatically generated feature-rich canyons.

Keywords: Procedural generation techniques; Terrain generation; Canyons; 3D Scenes; Mean Shift; Dijkstra's algorithm; Perlin noise.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MDE	Modelo Digital de Elevação
ESA	*Entertainment Software Association*
GPU	*Graphics Processing Unit*
GPGPU	*General-purpose Computing on Graphics Processing Units*
OpenCV	*Open Source Computer Vision*
OpenCL	*Open Computing Language*
UML	*Unified Modeling Language*
LaCA	Laboratório de Computação Aplicada da UFSM
PPA	*Profile recognition and Polygon breaking Algorithm*

SUMÁRIO

1 INTRODUÇÃO

A indústria de jogos de computador e video games movimenta bilhões de dólares por ano e emprega uma grande quantidade de pessoas. Segundo o relatório de 2010 da ESA (*Entertainment Software Association*) estima-se que essa indústria tenha faturado, somente nos Estados Unidos, em torno de 119.9 bilhões de dólares, entre 2005 e 2009. Durante esse mesmo período, essa indústria teve crescimento de 10.6% ao ano, enquanto que a economia americana teve crescimento anual de 1.4% (SIWEK, 2010).

Por outro lado, os jogos digitais vêm se tornando cada vez mais complexos, elaborados e apresentando uma enorme riqueza de detalhes gráficos. A força de trabalho necessária para a criação desses jogos tem que ser altamente capacitada, exigindo uma grande equipe de programadores, roteiristas, designers 2D e 3D, entre outros profissionais. Exemplificando o cenário em questão, a empresa *Polyphony Digital* gastou cerca de 60 milhões de dólares para a criação do jogo *Gran Turismo 5* (CROSSLEY, 2009).

Focando-se na área de artes gráficas 2D e 3D, técnicas procedurais vêm surgindo nas últimas três décadas como ferramenta para automatização da criação de parte deste conteúdo. De maneira geral, a geração procedural corresponde a síntese automática de conteúdo através de códigos de programação encapsulados em procedimentos (em inglês, *procedure*). Essas técnicas buscam auxiliar os artistas a criar terrenos, cidades, prédios, vegetação, efeitos e diversos outros modelos de maneira automatizada, diminuindo, com isso, a grande quantidade de horas necessárias para a criação de conteúdo. Dessa forma, é possível a criação de jogos mais atrativos a um custo mais reduzido.

EBERT et al. (1998) contextualiza o cenário da década de 80, no qual muitas técnicas procedurais de geração de texturas foram amplamente difundidas. Essas técnicas são capazes de gerar texturas realistas de madeira, mármore, pedras e outros materiais. Por outro lado, pesquisas recentes utilizam-se de diversas técnicas modernas para a criação de terrenos (HNAIDI et al., 2010; ZHOU et al., 2007; DORAN e PARBERRY, 2010; WALSH e GADE, 2010), costas (BEVILACQUA, 2009), cidades (GREUTER et al., 2003), ruas (GANG e GUANGSHUN, 2010; CHEN et al., 2008), estradas (GALIN et al., 2010) e rios (HUIJSER et al., 2010).

Como visto, a pesquisa em métodos procedurais é muito abrangente. Até mesmo a área de geração de terrenos, que apresenta um amadurecimento maior, tem poucos

trabalhos focados na geração de terrenos específicos. Durante a pesquisa bibliográfica, evidenciou-se a existência de poucos trabalhos focados especificamente na geração de cânions. Assim sendo, este trabalho desenvolve um método para a criação desse tipo de cenário (cânions) de forma automatizada com foco em jogos digitais, e que apresentem características como áreas de planícies, regiões de encosta suave, penhascos, regiões planas nas partes mais altas (planaltos) e um rio. Em todas as etapas envolvidas no processo de geração, usa-se imagens para a representação das informações. Sobre essas imagens são aplicadas operações de composição para se obter o mapa de altura final, que representa o cânion gerado.

Nesse contexto, a partir da experiência prévia do método *Mean Shift* (FUKUNAGA e HOSTETLER, 1975) aplicado ao processamento de imagens, notou-se que, quando executado sobre o ruído de Perlin (PERLIN, 1985), havia a formação clara de penhascos, que poderiam ser parametrizados para atender as questões referentes a formação dos cânions, dando assim continuidade aos trabalhos de geração procedural já desenvolvidos no LaCA (Laboratório de Computação Aplicada - UFSM), que tiveram início com o trabalho de BEVILACQUA (2009).

Além disso, esta dissertação está enquadrada dentro do contexto do projeto "Cluster Brasileiro de Visualização e Simulação Baseado em *Graphic Processor Units* (GPUs)". Esse projeto envolve a Universidade Federal Fluminense (UFF) e a Universidade Federal de Santa Maria (UFSM) e tem como órgão de fomento a CAPES. Dessa forma, utiliza-se técnicas que façam uso de processamento genérico em GPUs.

1.1 Definição do Problema: Cânions

Cânion, em geomorfologia, é a denominação que se dá a vales abruptos, profundos em forma de penhasco e, muitas vezes, escavado por um rio (GUERRA e GUERRA, 2008). Além disso, a Wikipedia (WIKIPEDIA, 2011) aprofunda mais a estrutura e características dos cânions conforme as características a seguir:

- A maioria dos cânions são formados por processo erosivos a partir de um nível de planalto;
- Os penhascos são formados devido a estratos de rochas duras que são resistentes a erosão e intemperismo. Estas rochas permanecem expostas nas paredes dos vales;

- O vento e a água do rio se combinam para erodir e cortar materiais menos resistentes.

Durante a análise das imagens avaliadas (ver fotos de cânions reais na seção 4.1) evidenciou-se que os cânions possuem, de forma geral, uma estrutura bastante semelhante, como exemplificado na Figura 1.1. Os processos erosivos, juntamente com as camadas que compõe o terreno, influenciam a forma dos cânions. A Figura 1.2 apresenta três cânions – *Grand Canyon*, *Canyon de Chelly* e o Itaimbezinho. O *Grand Canyon* destaca-se por apresentar a nítida formação de faixas esculpidas nas suas encostas como se fossem degraus. Por sua vez, o *Canyon de Chelly* apresenta grandes planícies envolto de paredões rochosos íngremes com as bases mais largas e com declive mais suave do restante do paredão. E, por fim, o Itaimbezinho apresenta a formação rochosa íngreme e com a ocorrência de vegetação nas suas paredes. A partir dessas observações destacam-se as seguintes características comumente encontradas em canions:

- Encostas com declive mais suave nas bases;
- Horizontalmente, existem faixas de transição entre camadas de terrenos que se repetem em todos os morros na mesma faixa de altura. Esta questão é principalmente referente à textura;
- Características de penhascos que se repetem em relação a mesma faixa de altura (estruturas que lembram degraus).

Figura 1.1: *Canyon de Chelly* (WELLER e WELLER, 2005), *Canyon de Chelly National Monument* (NEGAUNEE, 2009) e *Coal Mine Canyon* (COBALT123, 2010), respectivamente.

Com relação a geração procedural de cenários, GEORGE e HUGH (2006) enumeram uma série de características importantes utilizadas para avaliar trabalhos de geração procedural de cidades. As características apresentadas são genéricas e aplicáveis com relação

Figura 1.2: Da esquerda para a direita: *Grand Canyon*, *Canyon de Chelly* e o Itaimbezinho. Fotos respectivamente de THIERRY (2010), CURTIS (1904) e PIMENTA (2006).

ao contexto deste trabalho. Dessa forma, os critérios a seguir devem nortear a geração de procedural de cânions:

- Realismo — O cânion gerado deve se parecer com um cânion real;
- Escala — A escala do cânion gerado deve ser equivalente a do mundo real;
- Variações — A natureza apresenta uma infinidade de variações. Em outras palavras, as estruturas dos cânions devem apresentar diversas variações;
- Entrada — Define a menor quantidade de informação de entrada necessária para a criação de um cânion;
- Controle — Expressa como o usuário pode influenciar a criação dos cenários de cânion. Ou seja, se o método em questão é bastante assistido, pouco assistido ou não assistido.

As características mencionadas acima são tratadas ao longo da dissertação, sobretudo nos capítulos 3 (Algoritmo e Implementação) e 4 (Resultados).

1.2 Objetivos

O objetivo geral deste trabalho é a geração procedural de cenários 3D de cânions que sejam aplicáveis à jogos digitais. Os cenários gerados devem ser ricos em detalhes e caracterizados de forma que possam ser utilizados de maneira fácil por desenvolvedores de jogos. Dessa forma, dentre as características inerentes de terrenos importantes para jogos digitais, destaca-se que o terreno deve apresentar regiões acessíveis sem apresentar pontos cegos. Assim será possível movimentar um personagem, ou diversas unidades de

personagens, de um ponto ao outro no cenário. Para isso é importante que exista pelo menos um caminho que conecte as regiões de interesse. Em outras palavras, deve-se evitar a existência de regiões inatingíveis.

Como o trabalho mostrou-se bastante extenso é necessário especificar de maneira clara o que não faz parte do escopo proposto. Questões específicas como a geração de texturas e estruturas que apresentem reentrâncias, como cavernas ou negativos de penhascos, não fazem parte do escopo deste trabalho e são sugeridos como trabalhos futuros.

Outra questão que não se objetiva imitar é a formação geológica vista de uma visão aérea, por considerar que essa característica não crucial para muitos tipos de jogos, como, por exemplo, jogos de estratégia e de corrida.

A seção 4.1 apresenta, a fim de facilitar a avaliação dos resultados, diversas imagens da visão interna do cenário de cânions reais que se quer imitar. A seção 4.2, por sua vez, apresenta as imagens com características que estão fora do escopo desta dissertação.

1.2.1 Objetivos Específicos

A partir da avaliação de diversas imagens reais, evidencia-se um conjunto de características que se repetem. Dentre as características em comuns que observa-se entre grande parte dos cânions, a Figura 1.3 apresenta as principais questões numeradas de 1 à 6 a serem reproduzidas de forma procedural. Segue a descrição conforme a numeração da Figura 1.3 na lista a seguir:

1. Planalto – região de topo plana.
2. Penhasco – podendo ser composto também por diversos níveis;
3. Artefatos – são formações rochosas que lembram a forma de obeliscos e possuem variações de altura e tamanho;
4. Sopé da montanha – com declive suave;
5. Planícies – em alguns casos de cânions reais essa característica pode ser inexistente;
6. Rio – Normalmente apresenta o curso bastante acidentando com diversas curvas.

A partir das informações enumeradas, os objetivos específicos são:

- Utilizar ruído de Perlin para geração do reticulado inicial;

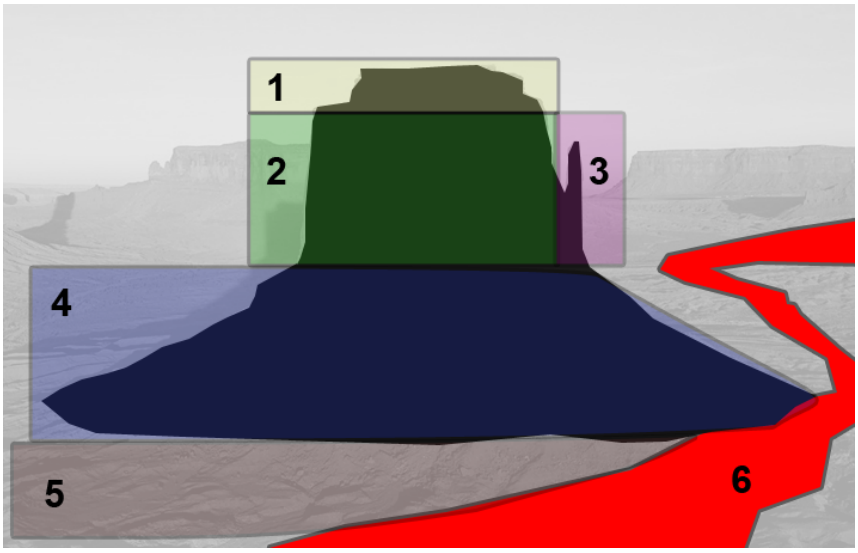

Figura 1.3: Diagrama proposto com as características base para a geração do cânion.

- Utilizar um algoritmo de clusterização, neste caso o *Mean Shift* (FUKUNAGA e HOSTETLER, 1975), para definir as estruturas básicas de Planaltos, Penhascos e Artefatos;
- Utilizar operação de segmentação por limiar para definir regiões de planícies;
- Utilizar algoritmo de rotulagem de componentes conectados (CHANG et al., 2004) para identificar as planícies;
- Utilizar o algoritmo de Dijkstra para garantir a conexão entre as planícies e para a definição do curso do rio;
- Aplicar um filtro espacial de suavização para definir um relevo suave no sopé da montanha.

1.3 Contribuições da dissertação

Este trabalho tem as seguintes contribuições: primeiro, através da análise empírica de um conjunto de imagens, identificou-se as principais características de um cânion no contexto da geração procedural. Dessa forma, propõe-se um diagrama base (Figura 1.3), que representa um corte transversal da estrutura montanhosa com as principais características de um cânion. O diagrama da Figura 1.4 apresenta a correlação das principais regiões identificadas, relacinando as estrutura da montanha com imagens de cânions reais.

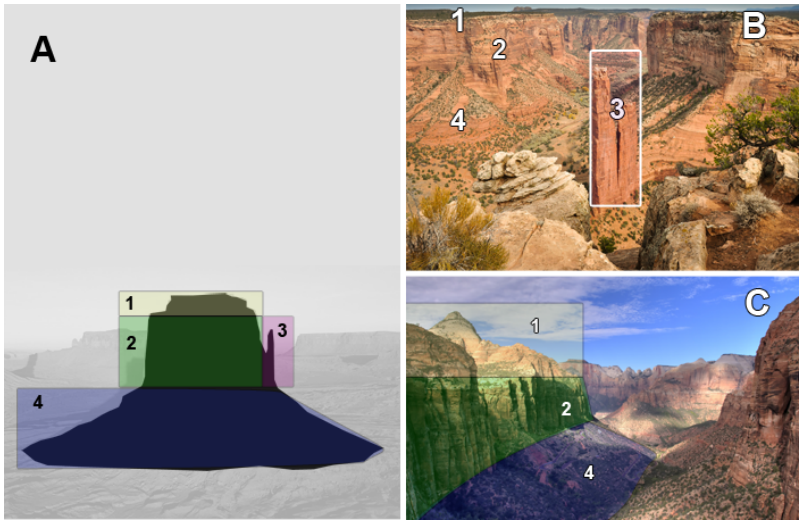

Figura 1.4: (A) Diagrama proposto para a geração da montanha: Base do relevo montanhoso dos cânions; (B) *Spider Rock Canyon* - Foto de HEY (2009a); (C) *Canyon Overlook* - Foto de LAU (2007a).

Outra questão, no contexto desta dissertação, refere-se sobre o uso do termo "recursivo". Neste trabalho este termo também é utilizado para denotar o padrão de auto replicação e similaridade que é identificado em alguns cânions. Dessa forma, a segunda contribuição refere-se a identificação da natureza recursiva dos cânions, conforme o diagrama da Figura 1.5. Nesse caso o número 1 corresponde ao planalto, 2 aos penhascos e 4 as regiões de declive suave. Nota-se que existe um empilhamento das regiões 2 e 4, caracterizando-se o princípio da recursão. A partir dessa constatação, podem-se desenvolver algoritmos recursivos que tirem vantagens da natureza de auto replicação dos cânions. Entretanto, devido a limitações de ordem temporal, este trabalho focou-se em desenvolver um algoritmo que pudesse gerar estruturas conforme o esquema da Figura 1.3, sem levar em conta estas características que se repetem de maneira recursiva.

A principal contribuição é a criação de uma técnica procedural, não recursiva e não assistida, para geração de cenários de cânions baseado em uma técnica de clusterização (*Mean Shift*), na rotulagem de componentes conectados e definição de rotas através de um conjunto de dados gerados pela técnica de ruído de Perlin.

Por se tratar de uma técnica completamente não assistida, ela apresenta uma grande vantagem com relação a compressão de informações. No caso de jogos *multiplayer*, somente é necessário enviar os parâmetros do cenário para todos os jogadores gerarem o mapa localmente. Em outras palavras, a criação do terreno pode ser realizada comple-

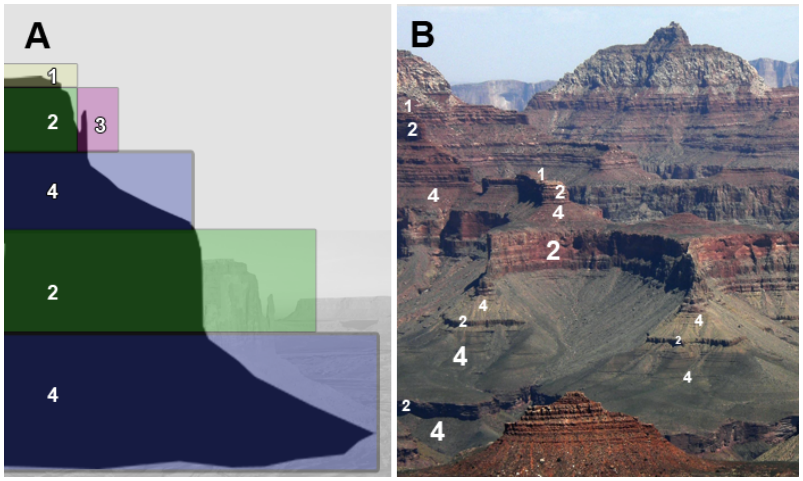

Figura 1.5: (A) Diagrama recursivo proposto para a geração da montanha: Relevo montanhoso dos cânions recursivo; (B) *Grand Canyon from South Rim near Visitor Center* - Foto de OLIVIER (2008).

tamente no cliente, o que evita o trafego de grande quantidade de informações de pela internet.

Outra contribuição desta dissertação é a introdução das técnica de clusterização (*Mean Shift*) e de rotulagem de componentes conectados para a área de geração procedural.

1.4 Organização da dissertação

O conteúdo desta dissertação está organizado em cinco capítulos. O segundo capítulo mostra os principais trabalhos e a revisão de literatura dos principais conceitos, métodos e algoritmos usados como base deste trabalho. O terceiro capítulo apresenta o algoritmo desenvolvido e questões de implementação. Os resultados são apresentados no capítulo 4. Por fim, no capítulo 5, apresenta-se as conclusões e trabalhos futuros.

2 REVISÃO DE LITERATURA

2.1 Trabalhos Relacionados

A geração procedural corresponde a síntese automatizada de algum tipo de conteúdo através de um determinado código. Ou seja, como o nome mesmo indica, através de *procedures*. A ideia básica é poder gerar, sempre que necessário, o conteúdo desejado. Nas últimas três décadas essas técnicas procedurais vêm sendo desenvolvidas para criar ou facilitar a criação de diversos tipos de conteúdo gráfico. A abordagem procedural busca auxiliar os artistas a criarem terrenos, estruturas, efeitos e diversos outros modelos de maneira automatizada. Diversas pesquisas atuais vêm sendo realizadas para a criação de relevos, prédios, cidades, ruas e rios. DE CARLI et al. (2011) apresenta, em um *survey*, os principais trabalhos nesta área e os classifica como assistidos e não assistidos.

No contexto desta dissertação, bem como no artigo de DE CARLI et al. (2011), métodos assistidos são aqueles que exigem forte intervenção humana a fim de atingir o resultado desejado. De forma oposta, métodos não assistidos são aqueles que exigem pouca ou nenhuma intervenção humana para guiar a construção do resultado desejado.

Para esta dissertação foram identificados três trabalhos de maior relevância que são descritos na sequência. São esses os trabalhos de BEVILACQUA et al. (2009), GALIN et al. (2010) e ZHOU et al. (2007). Além desses, alguns outros trabalhos mais antigos são apresentado para fins de uma melhor contexturalização da área de geração procedural.

Dentre os principais trabalhos relacionados, BEVILACQUA et al. (2009) apresentam Charack, uma ferramenta não assistida para geração de mundos pseudo-infinitos para jogos 3D. Apesar da abrangência geral de Charack, a sua principal contribuição está relacionada ao algoritmo de geração de costas, dessa forma, gerando praias, ilhas, baías e linhas costeiras que imitam formações costeiras do mundo real. Mesmo não retratando a geração de cânions, o trabalho de BEVILACQUA et al. (2009) se caracteriza por gerar cenários específicos para jogos de maneira completamente procedural, e esta é a principal característica em comum com esta dissertação.

Grande parte do algoritmo de geração da ferramenta Charack baseia-se na função de ruído de Perlin e em uma matriz global, chamada de macro matriz, gerada proceduralmente de maneira não assistida, para a definição das características do terreno em alto nível. Essa matriz informa se uma determinada região faz parte do relevo continental ou

costeiro. Além disso, essa matriz armazena meta informações que são utilizadas para a geração de cenários mais detalhados. Cada ponto do terreno é mapeado para uma entrada na matriz e ajustado em função da meta informação encontrada. Define-se a altura de cada ponto com base na descrição da macro matriz e de uma série de transformação e interpolações, que, por sua vez, influenciam os pontos vizinhos. É importante destacar que o tamanho dessa macro matriz é muito menor que o tamanho do terreno. Dessa forma, o algoritmo força a geração de conteúdo referente a cada ponto baseado, principalmente, na relação com seus vizinhos. Além disso, esse trabalho inspirou significativamente a solução proposta nesta dissertação pelo uso de ruído de Perlin como ponto de partida. A Figura 2.1 ilustra resultados gerados pela ferramenta Charack.

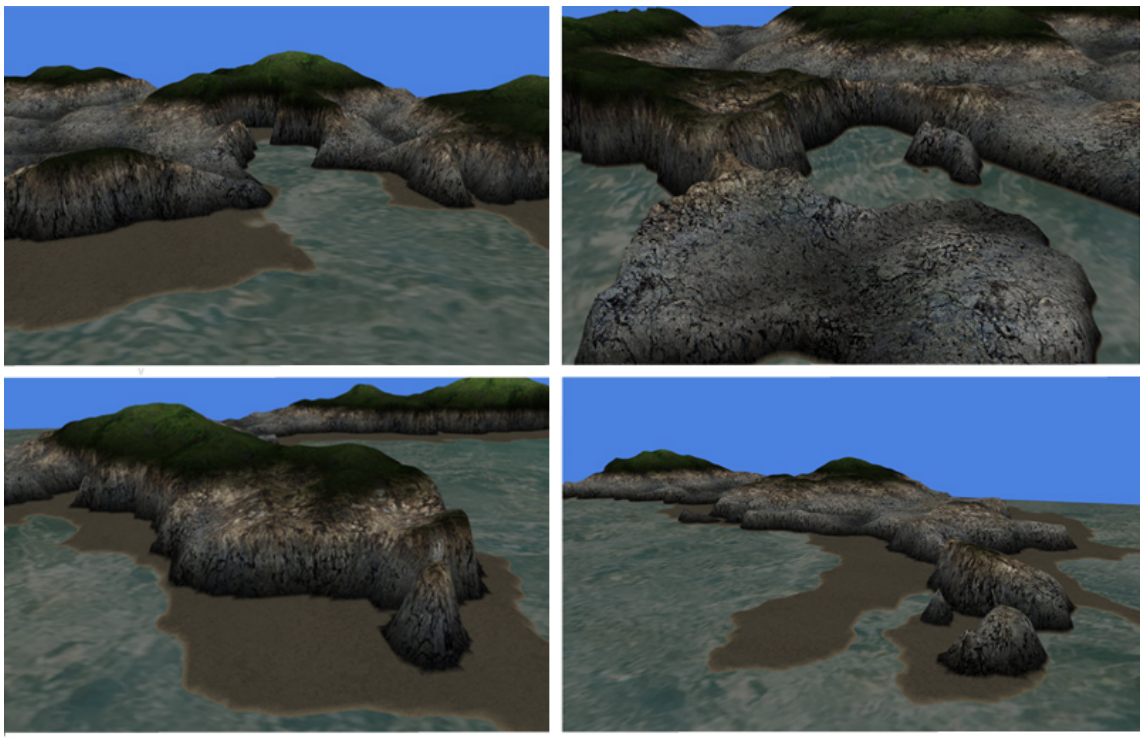

Figura 2.1: Costas geradas pela ferramenta Charack (BEVILACQUA, 2009).

GALIN et al. (2010) apresenta um método não assistido para geração de estradas baseado em um algoritmo anisotrópico ponderado de caminho mais curto. Dada uma cena de entrada, cria-se automaticamente um caminho que liga um ponto inicial a um final. A trajetória da estrada minimiza uma função custo que leva em conta os diferentes parâmetros da cena, incluindo a inclinação do terreno e obstáculos naturais como rios, lagos, montanhas e florestas. A estrada gerada pode apresentar pontes e túneis, de acordo com a complexidade dos obstáculos naturais.

Para o cálculo do caminho mais curto GALIN et al. (2010) usa o algoritmo A* (HART et al., 1968). Considerando-se uma região $\Omega \in R^2$ e dois pontos de entrada **a** (inicial) e **b** (final), o objetivo é calcular um caminho contínuo ρ de **a** até **b** que minimiza a integral de linha ao longo do caminho de uma função de custos ponderado $c(p, \dot{p}, \ddot{p})$. Essa função depende da posição p e de suas derivadas primeira e segunda, representadas por $\dot{p}$ e $\ddot{p}$, respectivamente. A função c de custo global proposta pode ser vista na equação 2.1. O conjunto de funções $k_i : R^3 \times R^3 \times R^3 \rightarrow R$ avalia as diferentes características do terreno e as características geométricas da trajetória da estrada no ponto p, sendo μ_i funções de transferência que pesam e combinam a influência das características $k_i(p, \dot{p}, \ddot{p})$. Estas funções de transferência permitem ao usuário controlar a influência relativa dos parâmetros da cena e, portanto, controlar a forma do caminho. A Figura 2.2 apresenta rotas geradas proceduralmente e influenciadas pelo ambiente.

$$c(p, \dot{p}, \ddot{p}) = \sum_{i=0}^{n-1} \mu_i \circ k_i(p, \dot{p}, \ddot{p}) \tag{2.1}$$

Figura 2.2: Estradas proceduralmente geradas e influenciadas pelo ambiente (rios, lagos, montanhas e florestas) (GALIN et al., 2010).

ZHOU et al. (2007) apresenta um trabalho de síntese de terrenos, a partir de modelos digitais de elevação. Nessa abordagem, partes de amostras do terreno já existentes são usadas para gerar um novo relevo. A síntese do terreno é guiada por um mapa de características que é esboçado pelo usuário. O algoritmo particiona esse esboço em pequenos fragmentos e procura no modelo digital de elevação (MDE), do terreno real, pelas características estruturais que correspondam à posição dos fragmentos no mapa esboçado. Tanto o modelo digital de elevação de entrada quanto o esboço do usuário são analisados utilizando-se técnicas da área de geomorfologia. Essa abordagem enfatiza características curvilíneas de grande escala, tais como rios e vales.

A extração das características – tais como vales, cumes, rios e morros – do MDE do

terreno real são obtidas usando um algoritmo adaptado de quebra de polígonos e reconhecimento de perfis (PPA) (em inglês, *"Profile recognition and Polygon breaking Algorithm"*) (SONG e HSU, 1998). A Figura 2.3 ilustra de forma resumida o algoritmo PPA. Após a extração das características, elas são combinadas no terreno resultante, de forma a casar com o esboço do usuário (ver figura 2.4). Por fim, uma suavização é aplicada entre os diferentes fragmentos, garantindo uma transição suave. As Figuras 2.5 e 2.6 ilustram a aplicação dessa técnica.

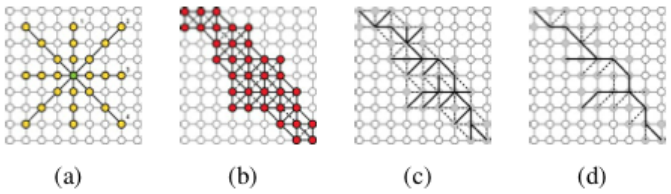

Figura 2.3: Algoritmo de quebra de polígonos e reconhecimento de perfis de SONG e HSU (1998): (a) Reconhecimento de perfil, (b) Conexão de destinos, (c) Quebra de polígonos e (d) Redução de ramos (ZHOU et al., 2007).

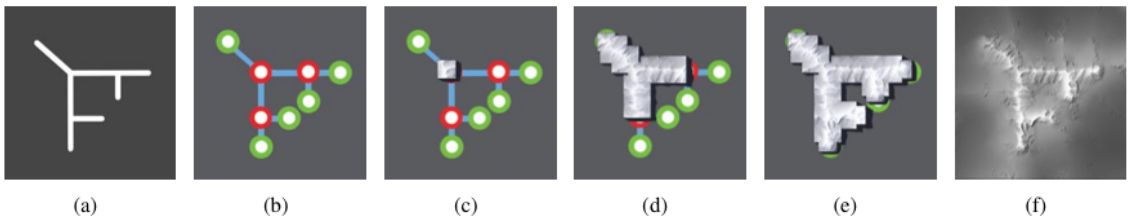

Figura 2.4: Ilustração da ordem da colocação das partes das características. (a) Exemplo de mapa esboçado; (b) Estrutura de árvore retornada pela análise do PPA; (c) A raiz é posicionada primeiro; (d) Um algoritmo de busca em largura guia a colocação das partes adicionais; (e) Uma vez percorrida a árvore, inicia-se a colocação das partes que não representam características; (f) Resultado final (ZHOU et al., 2007).

Dessa forma, esse trabalho é classificado como sendo um método assistido, visto que necessita da interação do usuário. Além disso, ele necessita de uma grande base de modelos de elevação digitais previamente existentes para poder gerar novos terrenos com grande variabilidade. Assim sendo, para poder gerar um cenário de cânion deve-se ter previamente o MDE de um terreno de cânion. Essa caracterísa acaba sendo uma das limitações do trabalho no contexto da geração procedural.

Dentre as diferenças principais do trabalho de ZHOU et al. (2007) com o desta dissertação destacam-se, primeiramente, que o método proposto nesta dissertação é completamente não assistido. Em segundo lugar, ele não necessita de modelos digitais de elevação pré-existentes.

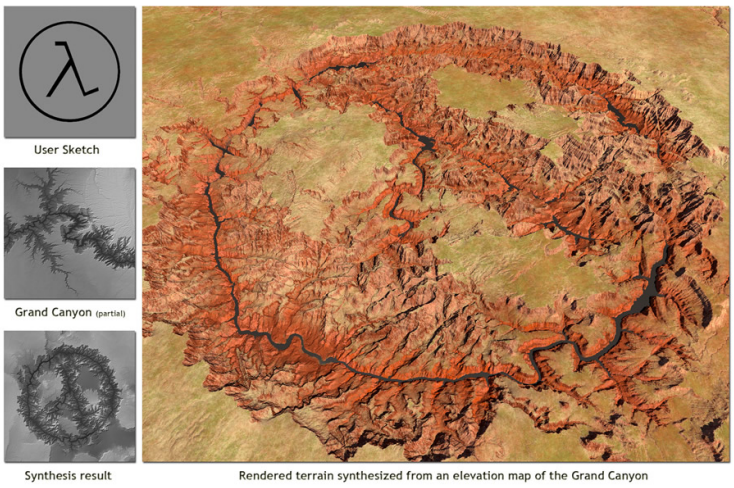

Figura 2.5: Terreno baseado no *Grand Canyon*. Na coluna da esquerda tem-se: o esboço do usuário, o MDE do *Grand Canyon* e o terreno sintetizado, respectivamente. À direita tem-se o terreno resultante em maiores detalhes (ZHOU et al., 2007).

Com o objetivo de ilustrar de maneira mais abrangente a área de síntese procedural, destacam-se, ainda, algumas técnicas antigas utilizadas na geração de terrenos que incluem o ruído de Perlin (PERLIN, 1985), *L-systems* (PRUSINKIEWICZ e LINDENMAYER, 1990) e fractais (MANDELBROT, 1977). Essas técnicas podem ser utilizadas sozinhas, combinadas entre si (DOLLINS, 2002; HÄGGSTRÖM, 2006; LINTERMANN e DEUSSEN, 1998; LINDA, 2007; PRUSINKIEWICZ e HAMMEL, 1993) ou, ainda, combinadas com outras técnicas como, por exemplo, os trabalhos de MUSGRAVE et al. (1989); OLSEN (2004); KELLEY et al. (1988); ROUDIER et al. (1993); CHIBA et al. (1998); NAGASHIMA (1998); BENES e FORSBACH (2001). A Figura 2.7 ilustra alguns desses trabalhos mencionados.

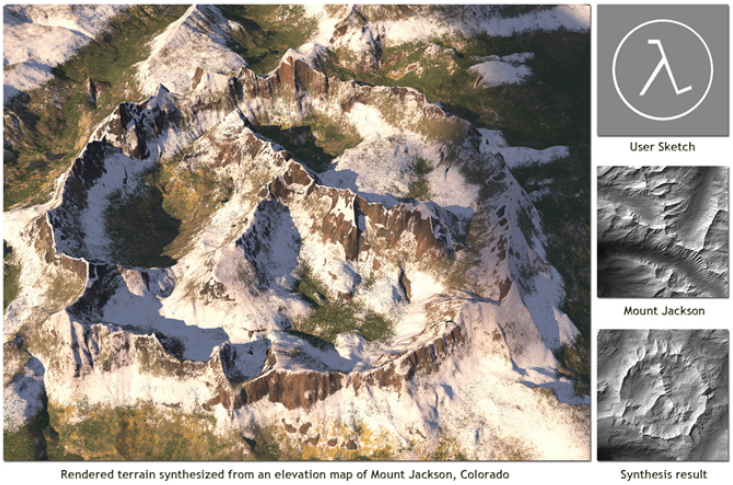

Figura 2.6: Terreno baseado no *Mount Jackson* (Colorado). Na coluna da direita tem-se: o esboço do usuário, o MDE do *Mount Jackson* e o terreno sintetizado, respectivamente. À esquerda tem-se o terreno resultante em maiores detalhes (ZHOU et al., 2007).

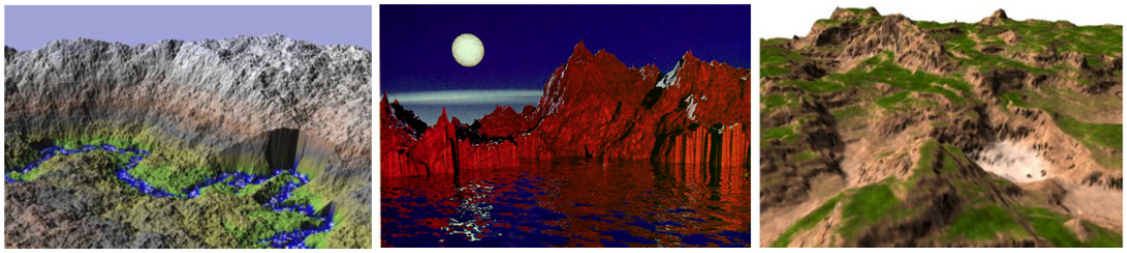

Figura 2.7: Trabalhos de PRUSINKIEWICZ e HAMMEL (1993), MUSGRAVE et al. (1989) e OLSEN (2004), respectivamente.

2.2 Mapa de Altura

Uma questão importante na área de terrenos é a sua forma de representação, a qual tem grande influência na maneira como o terreno é construído, as ferramentas para manipulá-lo e as características que consegue representar. Uma das maneiras mais usadas de representação é através de mapas de altura. Segundo GOMES e VELHO (1998), no universo matemático, o mapa de altura é um reticulado que define uma função apresentada na Equação 2.2, onde (x, y) são coordenadas do plano e z a altura correspondente.

$$F : U \subset R^2 \rightarrow R, z = f(x, y) \tag{2.2}$$

O mapa de altura pode ser visualizado como uma imagem em tons de cinza na qual os valores mais claros representam regiões mais altas (Figura 2.8 A) e, consequentemente, valores escuros as regiões mais baixas no terreno. A Figura 2.8 B apresenta uma grade triangular com a representação do relevo em um espaço tridimencional. Já a Figura 2.9

apresenta um exemplo prático desse tipo de representação.

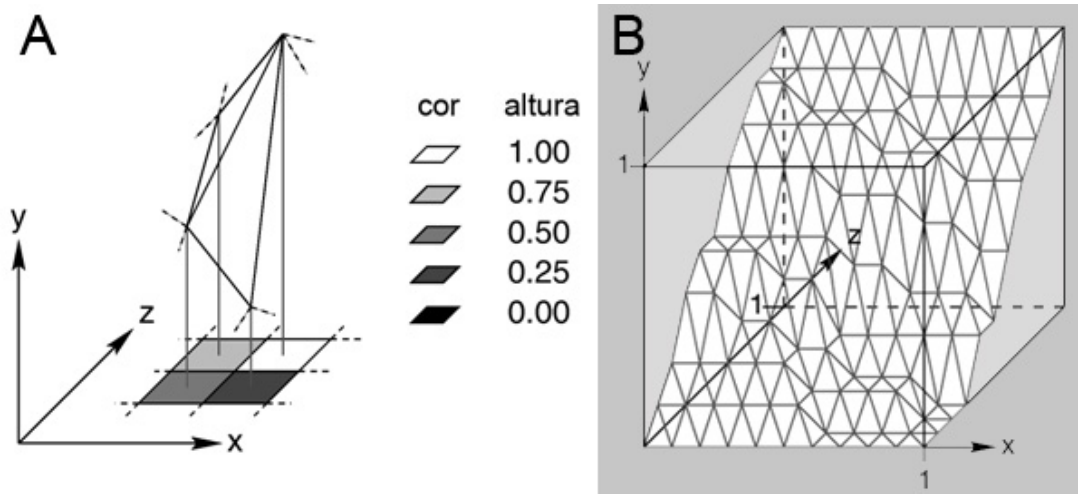

Figura 2.8: Mapa de altura. Adaptado de POV-RAY (2011).

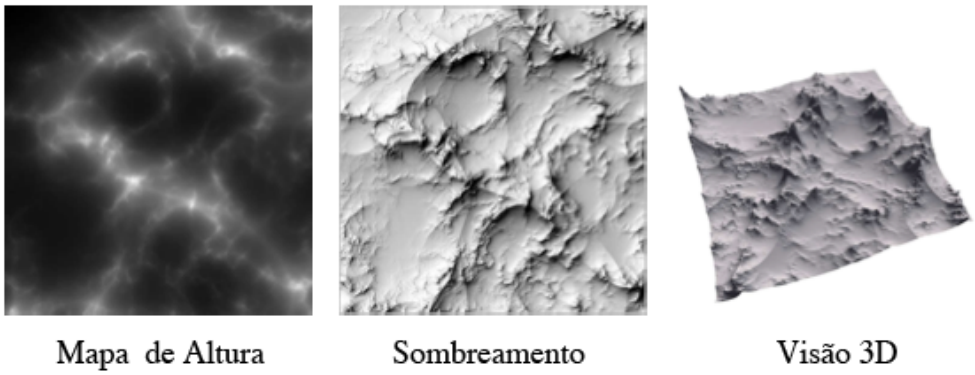

Figura 2.9: Representação de um mapa de altura. Adaptado de DACHSBACHER (2006).

Essa abordagem da representação de terrenos tem diversas vantagens, entre elas: a forma fácil e intuitiva de representação, facilidade de aplicação de algoritmos de detecção de caminhos (*Path Finding*) e detecção de colisões. Além disso, existe a possibilidade de operar diretamente sobre a imagem que representa o mapa de altura. Dessa forma, é possível utilizar ferramentas e técnicas de processamento de imagem para gerar o relevo com características desejadas. Em contrapartida, essa abordagem não consegue representar saliências, como arcos ou cavernas.

2.3 Ruído de Perlin

Segundo LAGAE et al. (2010), adicionar detalhes visuais de maneira eficiente em imagens sintéticas sempre tem sido um dos maiores desafios em computação gráfica.

O ruído procedural é uma das ferramentas mais bem sucedidas para a criação de tais detalhes.

Nesse contexto, o ruído de Perlin foi proposto em 1985 e, inicialmente, foi utilizado para gerar texturas realistas de materiais como nuvens, fogo, água, mármore, madeira e pedra (PERLIN, 1985). De maneira geral, o algoritmo corresponde a geração de funções de ruído individuais com amplitude e frequências diferentes, sendo a soma de todas o ruído de Perlin resultante (ELIAS, 2011). As Figuras 2.10 e 2.11 representam a geração do ruído de Perlin em uma e duas dimensões, respectivamente.

O ruído de Perlin usa o conceito de oitavas para definir as frequências das funções a serem somadas. Sendo assim, cada oitava é uma função de ruído que atua com o dobro da frequência da função anterior. Outros conceitos relacionados à persistência e interpolação podem ser vistos em maiores detalhes em ELIAS (2011). De maneira geral o ruído de Perlin é responsável por influenciar grande parte dos trabalhos relacionados à geração procedural, sendo considerada a primeira função de ruído procedural (LAGAE et al., 2010).

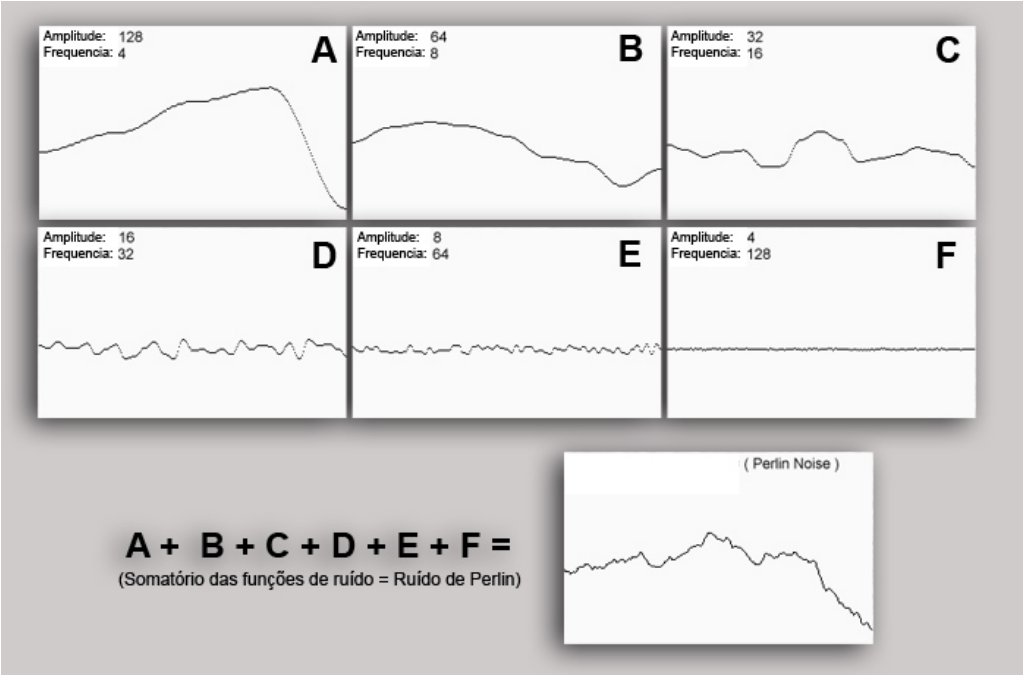

Figura 2.10: Em uma dimensão: diversas funções com amplitude e frequência somadas resulta no ruído de Perlin. Adaptado de ELIAS (2011).

Esta dissertação faz uso do ruído de Perlin como ponto de partida para a geração de um reticulado inicial, que é manipulado a fim de representar as características do cenário pretendido, como é apresentado no capítulo 3. Diversas outras técnicas poderiam ser usadas

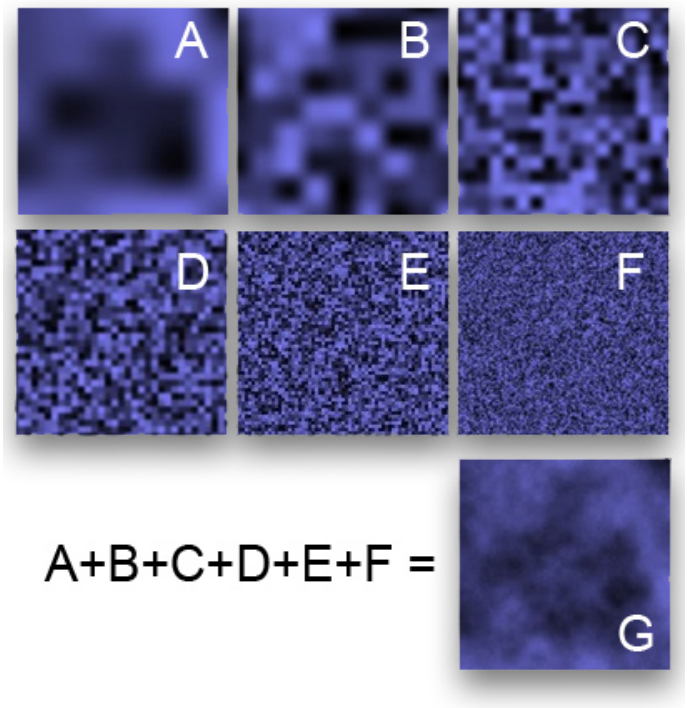

Figura 2.11: Em duas dimensões: diversas funções com amplitude e frequência somadas, sendo **(G)** o ruído de Perlin resultante da soma. Adaptado de ELIAS (2011).

com o intuito de gerar esse reticulado inicial. Contudo, devido a resultados promissores apresentados no início da pesquisa, ao se usar a técnica do ruído de Perlin, optou-se, então, pelo uso desse algoritmo. Além disso, a técnica proposta por Perlin caracteriza-se pelo baixo custo computacional e facilidade de parametrização. Soma-se, ainda, a limitação de tempo para a realização desta pesquisa. Dessa forma, sugere-se como trabalho futuro avaliar outras técnicas para a geração do reticulado inicial, comparando-as com o resultado obtido com o uso do ruído de Perlin. BEVILACQUA (2009) apresenta algumas dessas técnicas que potencialmente podem ser utilizadas para a geração do reticulado em questão, sendo elas fractais, divisões estocásticas, falhas geológicas, deposição de sedimentos e disposição do ponto médio.

A implementação do ruído de Perlin utilizada para esta dissertação foi desenvolvida na linguagem C por Ken Perlin, tendo sido publicada no CD que acompanha o livro *"Texturing and Modeling: A Procedural Approach"* (EBERT et al., 1998).

2.4 Clusterização de Dados

A clusterização de dados é responsável por separar os dados em grupos, também conhecidos como *clusters*. Esses *clusters*, por si só, podem ser utizados para interpretar

um conjunto de informações relevantes sobre um assunto de qualquer natureza. Outra utilidade para o agrupamento em questão está associado a segmentação de imagens, a qual pode servir de ponto de partida para várias aplicações. A fim de exemplificação, na Figura 2.12 apresenta-se diferentes agrupamentos de dados sobre o mesmo conjunto de pontos. Para isso, usa-se algum tipo de medida de similaridade.

No caso da clusterização que se objetiva extrair significado, o trabalho de MARTINS (2012), por exemplo, busca a classificação de fases em imagens hiperespectrais de raio X característico através do método *Mean Shift*.

Por outro lado, nesta dissertação usa-se a clusterização como ponto de partida para outras técnicas: o processo é utilizado como etapa necessária para a construção dos penhascos, planícies, encostas e planaltos do método de geração de cenários de cânions.

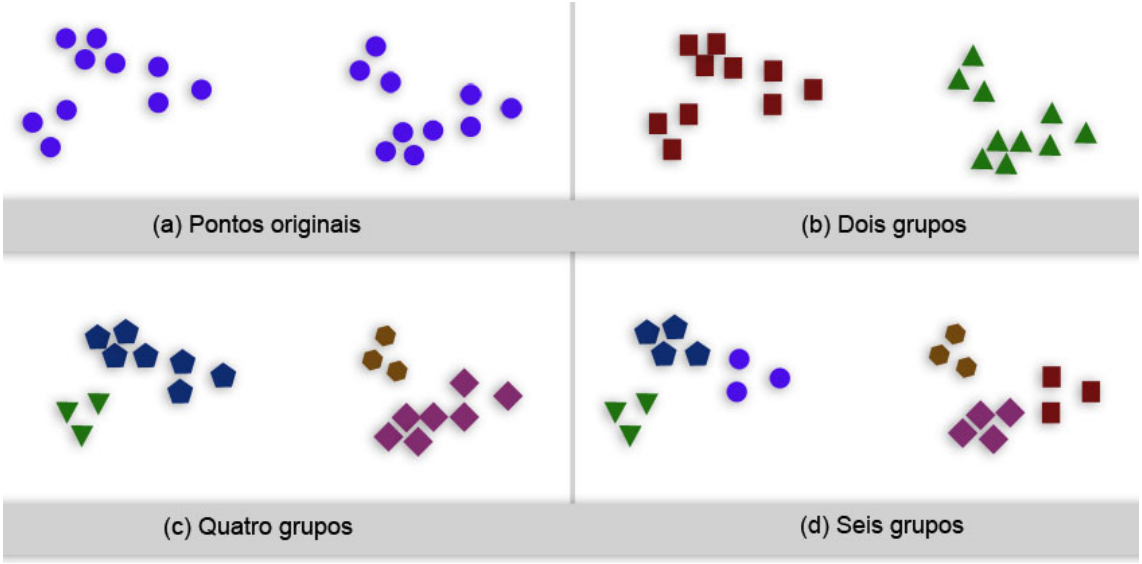

Figura 2.12: Diferentes maneiras de clusterização do mesmo conjunto de pontos. Adaptada de TAN et al. (2009).

2.4.1 Algoritmo *Mean Shift*

O *Mean Shift* é um método de clusterização de dados, não supervisionado, introduzido na década de 70 por Fukunaga e Hostetler (FUKUNAGA e HOSTETLER, 1975). Mais tarde, CHENG (1995) mostra uma generalização adequada do método, bem como sua complexidade algorítmica de $O(n^2)$, onde n é o número de elementos do conjunto de dados. Cheng também afirma que é possível reduzir essa complexidade para $O(n\ log\ n)$. Ou seja, o método *Mean Shift* é bastante complexo no uso dos recursos computacionais e a utilização de técnicas modernas de paralelização com GPGPU pode melhorar o seu

desempenho.

O procedimento do *Mean Shift* localiza os máximos de uma função densidade através do conjunto de dados discretos amostrados dessa função (CHENG, 1995). De forma geral o algoritmo funciona da seguinte maneira:

- Para cada ponto, delimita-se uma região, ou raio h, ao seu redor;
- Dentro da região delimitada, calcula-se o centro de massa;
- Desloca-se a região a ser calculada para o centro de massa, ou centróide, encontrado e repete-se o algoritmo até a convergência.

Em outras palavras, as regiões do espaço de características, onde a densidade de pontos é mais alta, correspondem aos *clusters*, sendo isto as modas da função densidade $f(x)$. Contudo, a função densidade não é previamente conhecida. Dessa forma, estima-se a função a partir de amostras. A função densidade estimada, $\hat{f}(\vec{x})$ (Equação 2.3), pode ser encontrada através do uso do gradiente. Assim sendo, a busca dos *clusters* pode ser feita pelo cálculo do gradiente da densidade estimada $\nabla\hat{f}(\vec{x})$ (Equação 2.4) (ROSA, 2008). Outra questão importante a ser destacada é que a função densidade estimada tem relação direta com o raio h. A escolha do raio tem grande impacto sobre o número de *clusters* que o *Mean Shift* identifica.

$$\hat{f}(\vec{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\vec{x}-\vec{x_i}}{h}\right) \tag{2.3}$$

Uso de um *kernel* K nos permite a generalização da equação quando o espaço de características não for um espaço Euclidiano. Ou seja, em um espaço de características d-dimensional a relação de distância entre os pontos pode levar em consideração pesos diferentes a dimensões diferentes. Segundo COMANICIU e MEER (1999), o uso de um *kernel* diferenciável permite que a estimativa do gradiente da densidade seja definida como o gradiente do kernel estimador de densidade conforme a equação 2.4.

$$\nabla\hat{f}(\vec{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} \nabla K\left(\frac{\vec{x}-\vec{x_i}}{h}\right) \tag{2.4}$$

Dando prosseguimento, a equação 2.5 define o vetor de *mean shift*. A qual foi apresentada por FUKUNAGA e HOSTETLER (1975).

$$M_h(\vec{x}) = \frac{1}{k} \sum_{x \in S_h(\vec{x})} \vec{x_i} - \vec{x} \tag{2.5}$$

ROSA (2008) explica que a equação 2.5 é a média dos pontos amostrados dentro de uma hiperesfera $S_h(x)$ de raio h e com centro em x, onde k corresponde ao número de pontos pertencentes à hiperesfera $S_h(x)$, sendo essa média correspondente ao centro de massa da hiperesfera. O *Mean Shift* é a diferença entre a média da amostra em questão e o ponto x.

Conforme mencionado anteriormente, a escolha do valor do raio h tem influência direta sobre a quantidade de *clusters* que o algoritmo irá identificar. No contexto de reconhecimento de padrões para microscopia, por exemplo, esses *clusters* podem corresponder a elementos de características diferentes. Ou seja, dependendo da escolha do raio, o algoritmo poderá identificar um grupo com X ou Y elementos, o que pode representar um problema, visto que essa diferença poderá representar um peso significativo na interpretação dos dados. Para um melhor entendimento, MARTINS (2012) apresenta a influência da escolha do raio h com relação a classificação de fases em imagens hiperespectrais de raio X característico com o algoritmo do *Mean Shift*.

No contexto desta dissertação, o *Mean Shift* atua em um espaço de características unidimensional, na qual a hiperesfera h acaba sendo um segmento nesse espaço de uma dimensão. A única característica levada em conta é a altura de cada coordenada pertencente ao reticulado. Assim sendo, a distância representada por h não se refere ao quão próximo os pixels estão uns dos outros no reticulados, mas sim quão próximo estão no espaço de características. Em outras palavras, quando maior a proximidade dos valores de intensidade dos pixels, maior é a chance deles pertencerem ao mesmo cluster. Dessa forma, o raio h funciona no sentido de definir a quantidade de penhascos que o algoritmo de geração de cânions irá gerar. O que acontece é que o resultado obtido é ponto de partida para a construção dos paredões, planícies e encostas suaves. Dessa forma, quanto menor o raio h maior será a quantidade segmentos e, por consequência, menos abruptos serão os penhascos.

Além do *Mean Shift*, o *k-means* (MACQUEEN, 1967) é outro método de clusterização bastante utilizado. Esse, por sua vez, classifica um determinado conjunto de dados baseado em um número conhecido de clusters. Ou seja, para classificar os dados é necessário conhecer o número de clustes, que é representado por *k*. A ideia principal é definir

os *k* centróides, sendo um para cada cluster. Um ponto a se destacar nesse método é que os centróides devem ser posicionados com bastante cuidado, pois posições diferentes influenciam completamente o resultado da clusterização. Dessa forma, a melhor escolha é posicionar o centróide o mais distante um dos outros. O passo seguinte é associar cada ponto no espaço de características com o centróide mais próximo. A partir desse momento, calcula-se novos centróides *k*, deslocando-os para o centro de massa dos pontos previamente clusterizados. O próximo passo é associar os pontos novamente com o novo cluster *k* mais próximo. Recalcula-se novamente o centróides *k* até o processo convergir.

De maneira geral, o algoritmo *k-means* é menos automatizado que o *Mean Shift*, exigindo, assim, maior interação com o usuário. Ou seja, no primeiro caso é necessário conhecer o número de clustes, bem como ter a estimativa dos centróides *k* que correspondem a cada um desse clusters, para ser possível, assim, definir os pontos pertencentes a cada cluster. Já o *Mean Shift* consegue fazer essa definição baseando-se apenas na parametrização do raio *h*. Ao se considerar o objetivo principal desta dissertação, que é desenvolver um método de geração procedural de conteúdo de maneira automatizada, fica evidente a necessidade do uso de algoritmos mais automatizados. Dessa forma, optou-se pelo pelo uso do algoritmo *Mean Shift* ao invés do *k-means*.

Para esse trabalho, foi utilizada uma implementação do *Mean Shift* baseada no algoritmo implementado por FELDMAN (2006), em *Matlab*, sendo que o código usado foi desenvolvido no LaCA (Laboratório de Computação Aplicada) da Universidade Federal de Santa Maria em C++ utilizando *OpenCL* para a aceleração em *hardware*.

2.5 Rotulagem de componentes conectados

No contexto da área de visão computacional é comum o emprego de técnicas para extrair informações de objetos em imagens, como tamanho, geometria, forma e posição. As técnicas de rotulagem de componentes conectados permitem identificar esses objetos em imagens binárias e extrair boa parte dessas informações. No caso desses algoritmos não retornarem todas as informações necessárias, elas acabam sendo utilizadas como uma etapa em um algoritmo mais complexo.

SHAPIRO e STOCKMAN (2001) apresentam rotulagem de componentes conectados da seguinte maneira:

> Considerando que B seja uma imagem binária e que $B[r, c] = B[r', c'] = \upsilon$ onde $\upsilon = 0$ ou $\upsilon = 1$. Um pixel $[r, c]$ está conectado ao pixel $[r', c']$ com relação ao valor υ se existir uma sequência de pixels $[r, c] =$

$[r_0, c_0], [r_1, c_1], ..., [r_n, c_n] = [r', c']$ em que $B[r_i, C_i] = \upsilon$, $i = 0, ..., n$, e $[r_i, c_i]$ vizinhos $[r_{i-1}, c_{i-1}]$ para cada $i = 1, ..., n$. A sequência de pixels $[r_0, c_0], ..., [r_n, c_n]$ forma um caminho conectado de $[r, c]$ até $[r', c']$.

A fim de ilustrar a definição dada por SHAPIRO e STOCKMAN (2001), a Figura 2.13 apresenta uma imagem binária com cinco componentes conectados de 1.

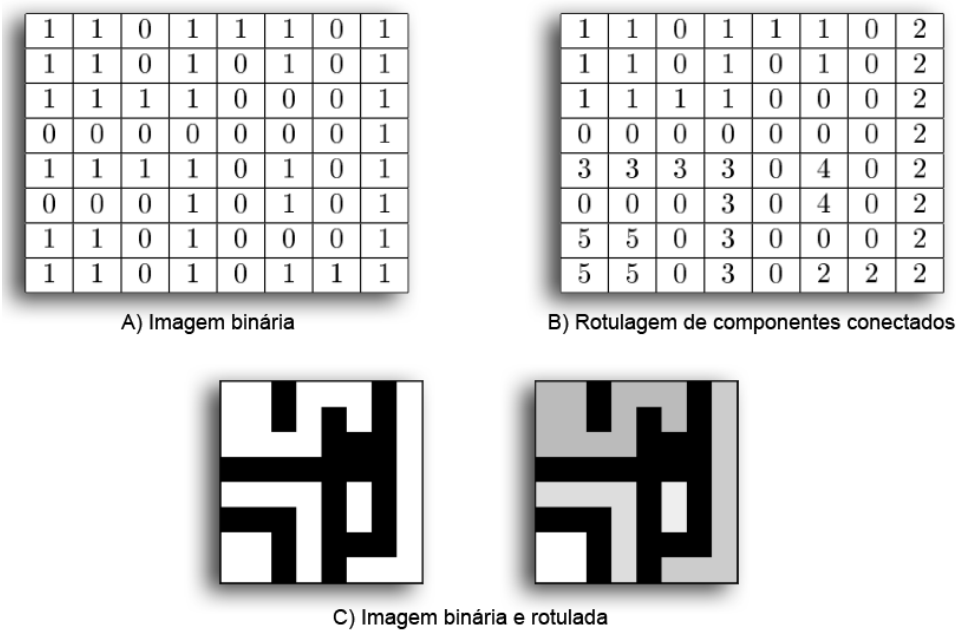

Figura 2.13: Imagem binária a ser rotulada (A); representação objetos conectados rotulados (B); imagem biária antes e após o processo de rotulagem, respectivamente (C). Adaptado de SHAPIRO e STOCKMAN (2001).

De maneira mais contextualizada com esta dissertação, CHANG et al. (2004) apresenta um algoritmo de tempo linear para rotulagem de componentes conectados, baseado em técnica de rastreamento de contorno (em inglês, *a linear-time component-labeling algorithm using contour tracing technique*). Nesse contexto, a biblioteca cvBlobs (CVBLOBS, 2012) implementa o algorimto de CHANG et al. (2004) e é utilizada para a implementação da solução desta dissertação. Dentre as vantagens da implementação em questão, destacam-se as estruturas de dados associadas, como uma lista encadeada dos centróides de cada objeto encontrado. Outro ponto relevante é poder associar uma heurística para rotular somente objetos que possuam uma quantidade mínima de pixels. Além disso, a implementação em questão foi feita em C++ dentro do padrão de programação do OpenCV. Ou seja, uma técnica eficiente dentro do contexto das tecnologias escolhidas.

Assim sendo, esta dissertação utiliza a técnica de rotulagem de componentes conec-

tados de CHANG et al. (2004) no contexto de identificar as planícies e, a partir das suas posições (centróides), aplicar o algoritmo de Dijkstra a fim de garantir as conexões entre elas. Um dos pontos chaves desta dissertação é permitir que o cenário seja utilizado para jogos digitais. Tomando-se por base os jogos de estratégia, garantir a conexão entre planícies é fator desejável. Para exemplificar essa questão, pode-se imaginar que um jogador tenha um grupo de infantaria que deseja deslocar à base do seu oponente, o que demanda a existência de pelo menos um caminho para permitir tal manobra.

2.6 Algoritmo de Dijkstra

O algoritmo de Dijkstra (DIJKSTRA, 1959) foi criado em 1959 pelo cientista da computação holandês Edsger Wybe Dijkstra. Esse algoritmo tem por objetivo encontrar o caminho mais curto entre dois vértices de um grafo, sendo que a busca no grafo é realizada em formato radial. Além disso, segundo MARIANI (2012), o algoritmo em questão não garante a exatidão da solução caso haja a presença de arcos com valores negativos.

Outro algoritmo bastante conhecido é o A* (HART et al., 1968), que é muito semelhante ao Dijkstra. Porem, além da função de custo acumulado $g(x)$ (equação 2.6), também usa uma função heurística $h(x)$ que estima a distância do nó de busca atual ao nó destino. Diferentemente do Dijkstra, para o A* deve-se saber onde está o nó a ser buscado. O uso dessa estimativa permite ao A* definir uma direção e indicar quais nós devem ser testados. Isso evita que muitos nós desnecessários tenham que ser visitados, otimizando a abordagem de Dijkstra em muitos casos.

A função heurística nunca pode superestimar o custo do nó ao destino. Dessa forma, o custo estimado deve ser menor ou igual ao custo real. O peso relacionado a heurística define o quão rápido e preciso o A* vai ser. No caso do algoritmo de Dijkstra a função heurística $h(x)$ não existe ou sempre assume o valor 0.

$$f(x) = g(x) + h(x) \tag{2.6}$$

Apesar do A* ser mais eficiente, nesta dissertação adotou-se o algoritmo de Dijkstra, visto que, como apresentado na seção 3.4, o grafo de navegação pode apresentar pesos negativos, o que tornaria muito complexo a criação de uma heurística válida para esse problema. Ambos os algoritmos não foram projetados para operar com pesos negativos, porém não se objetiva encontra a rota mais curta, mas sim uma rota que melhor retrate

as características de rios presentes em cânions. Além disso, os grafos de navegação presentes nos cenários não possuem ciclos negativos, o que sim inviabilizaria o uso desses algoritmos nesta dissertação.

3 ALGORITMO E IMPLEMENTAÇÃO

Neste capítulo, apresenta-se a solução desenvolvida, a qual é uma abordagem não assistida para criar cenários de cânions, representando-os na forma de mapas de alturas. A Figura 3.1 apresenta, na direita, o resultado do cenário do cânion atingido pela abordagem desenvolvida. Essa abordagem é realizada a partir do processamento do reticulado inicial criado com a técnica de ruído de Perlin, na esquerda.

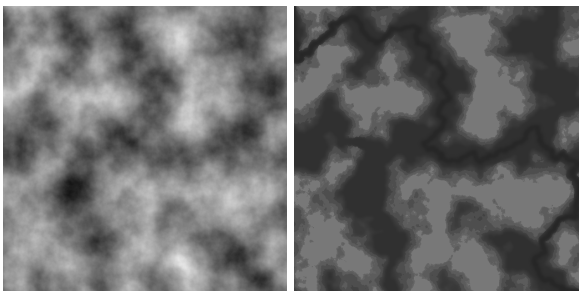

Figura 3.1: Na esquerda, Ruído de Perlin; na direita, Mapa de Altura de um cânion gerado proceduralmente.

Para gerar o cenário do cânion 3D, o algoritmo leva em conta diversas características observadas em cânions reais – já apresentadas nos diagramas das Figuras 1.3 e 1.4. Dessa forma, os principais pontos a serem tratados pela solução proposta nesta dissertação são definidos em função da existência de planaltos, penhascos, artefatos, sopé da montanha, planícies e o rio. O diagrama de atividades UML da Figura 3.2 apresenta os principais passos que foram implementados para a geração do relevo.

A divisão da solução em atividades, baseadas nas características que se quer replicar, apresenta vantagens. Primeiramente, ela consegue representar a estrutura da solução de forma intuitiva. O segundo ponto é que essa organização permite a criação e a substituição de partes do algoritmo de maneira fácil e com pouco impacto na solução global. Ou seja, caso seja desenvolvida uma nova técnica para a definição de planícies, por exemplo, essa poderia substituir a lógica anterior sem causar prejuízo no restante do algoritmo.

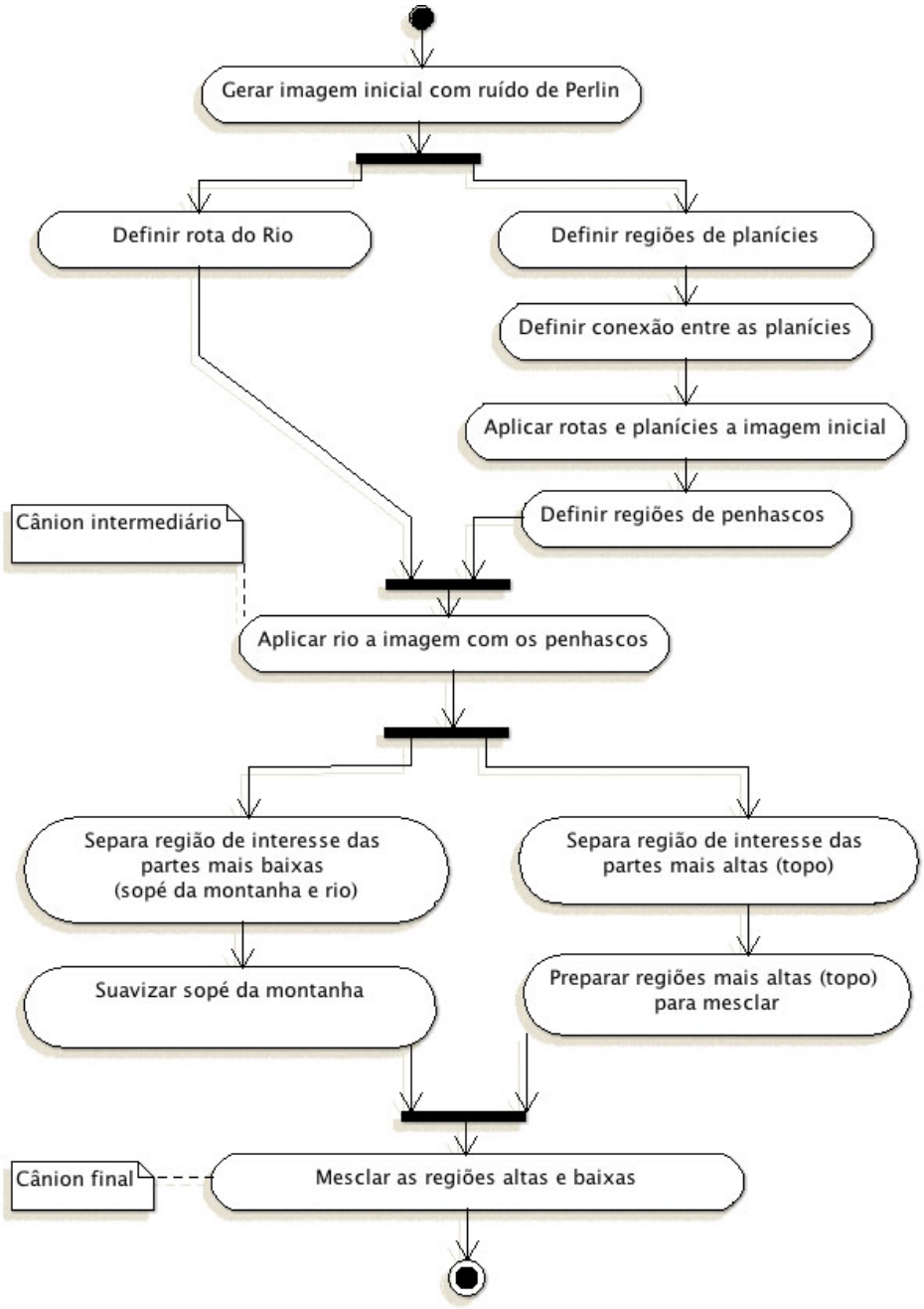

Figura 3.2: Diagrama UML de atividades: geração procedural de cenários de cânions.

3.1 Algoritmo

O algoritmo, em alto nível, é composto por um conjunto de técnicas que, quando combinadas, resultam no cenário desejado. As técnicas utilizadas são o ruído de Perlin, *Mean Shift*, rotulagem de componentes conectados e o algoritmo de Dijkstra.

O ponto de partida é a geração de um reticulado inicial, que será manipulado a fim de criar o terreno desejado, como é apresentado na Figura 3.1. Para isso, utiliza-se a técnica de ruído de Perlin. O uso desse ruído, por si só, não gera terrenos com características de cânions. Contudo, ao se manipular o reticulado utilizando-se de uma combinação de técnicas é possível gerar mapas de alturas com as características desejadas.

O próximo passo é a definição das regiões de planícies, ou seja, as regiões planas mais baixas do terreno e que são de grande interesse para serem explorados por desenvolvedores de jogos. Contextualizando com jogos de estratégia, essas planícies devem permitir confrontos entre os participantes, além de permitir a construção de bases. A única área mais baixa que a região das planícies é o curso do rio, sendo que este leva em conta a altitude das planícies. Diversas abordagens podem ser utilizadas a nível de implementação para a definição dessa região – como é apresentada a nível de implementação na seção 3.3. Contudo, a ideia básica é definir um determinado valor que sirva de limiar para a identificação das regiões mais baixas. Esse valor em questão, apesar de parecer um valor arbitrário, é uma importante forma de controle do usuário. Ou seja, através dessa definição é possível criar terrenos com mais ou menos planícies. A Figura 3.3 apresenta a máscara binária contendo as regiões de planícies, representadas pelas partes brancas, e as outras regiões, representadas pela parte escura. A equação 3.1 define as planícies.

$$destino(x,y) = \begin{cases} 1 & se\ img(x,y) < limiar \\ 0 & \text{caso contrário} \end{cases} \quad (3.1)$$

De posse das planícies, o passo seguinte é extrair as informações referentes as suas posições no reticulado. Como previamente comentado, as planícies devem apresentar rotas que as conectem entre si para evitar a existência de pontos cegos. Para isso, utiliza-se uma técnica de rotulagem de componentes conectados a fim de identificar todas as planícies. Com o resultado dessa rotulagem é possível obter o centróide correspondente a cada planície. Assim sendo, são criadas rotas artificiais para conectar essas planícies através dos seus centróides. A abordagem usada para definir as conexões entre essas regiões, bem como para a definição do rio, é baseada no princípio da mínima ação, e

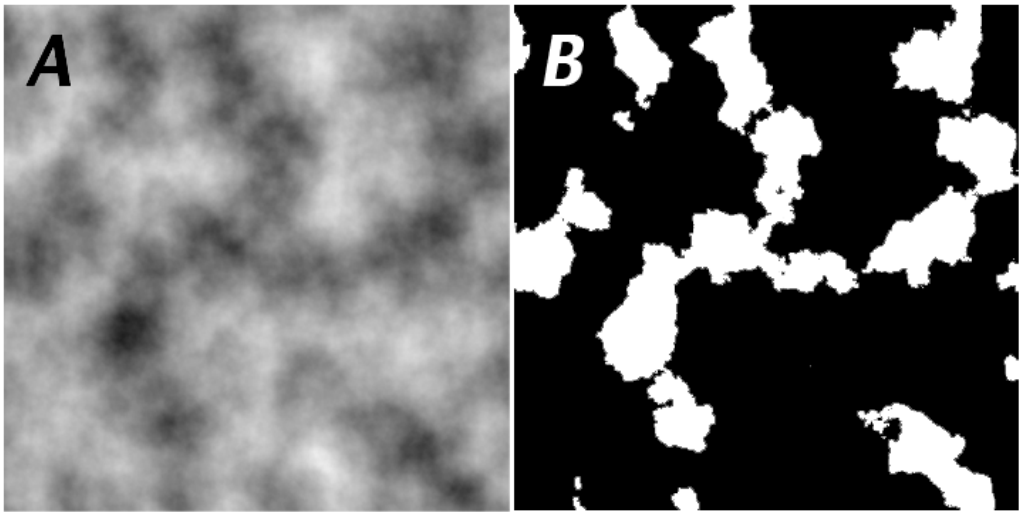

Figura 3.3: (A) Mapa de altura inicial gerado pelo ruído de Perlin. (B) Máscara binária contendo as regiões de planícies.

utiliza o algoritmo de Dijkstra – essa técnica é explicada detalhadamente na seção 3.4. É importante ressaltar que as coordenadas inicial e final do rio são definidas pelo usuário, da mesma forma que as outras parametrizações, o que permite maior controle sobre o terreno gerado.

É importante destacar que tanto as rotas que conectam as planícies, quanto as próprias planícies, assumem o valor do limiar previamente mencionado como suas alturas. Para a definição da altura do rio, é importante que esse valor seja inferior ao limiar. Dessa forma, o rio será sempre mais baixo que as planícies. Uma opção seria, simplesmente, assumir o valor 0 para a definição do curso do rio ou, como no caso desta dissertação, definir a elevação da região mais profunda do rio como sendo metade da intensidade do limiar escolhido.

Até este momento já se tem as planícies, as rotas e o curso do rio. O próximo passo está relacionado a construção dos precipícios sendo, então, utilizado o algoritmo *Mean Shift*. Objetivando-se utilizar o método de clusterização, aplicam-se as regiões de planícies e suas respectivas conexões sobre o reticulado inicial, como é mostrado na equação 3.2. Em outras palavras, as únicas regiões que permanecem inalteradas com relação ao reticulado original são as representadas pelas coordenadas de intensidade 0. Para todas as outras regiões, assumem-se a intensidade do limiar comentado anteriormente. Essa aplicação das planícies e rotas é a precondição para a definição dos penhascos através do algoritmo *Mean Shift*. A aplicação do rio segue a mesma ideia, mas é realizada posteri-

ormente ao *Mean Shift* para evitar que o curso do rio seja agrupado juntamente com as regiões de planícies.

$$destino(x,y) = \begin{cases} img(x,y) & \text{se máscara\_binária(x,y)} = 0 \\ limiar & \text{caso contrário} \end{cases} \quad (3.2)$$

A Figura 3.4 apresenta em maiores detalhes como a técnica de clusterização funciona no espaço de características para a definição das regiões de penhascos. É importante frisar que o *Mean Shift* não atua com relação as posições relativas de cada pixel, mas sim baseia-se unicamente em sua intensidade. O eixo v, dessa forma, representa um vetor unidimensional contendo todos os pixels da imagem. O eixo z, por sua vez, representa a intensidade de cada pixel. Na figura em questão, na letra (A), tem-se apenas a representação das alturas dos pixels em um dado espaço de características. Na letra (B) tem-se a representação dos pixels e as regiões dos clusters a que pertencem, sendo que cada cluster é representado uma cor diferente. Nesse esquema, é possível identificar facilmente a formação de regiões planas, pois todos os pixels pertencentes a um mesmo cluster assumirão a mesma intensidade. A letra (C) apresenta apenas o resultado da clusterização e isso já permite, quando renderizado em um espaço 3D, identificar os penhascos formados. Contudo, ainda falta a aplicação do curso do rio, que é realizada através da equação 3.2 usando-se apenas metade da intensidade do limiar das planícies. Neste momento, tem-se um cânion intermediário.

Agora já é possível identificar os penhascos, as planícies e o curso do rio. Basicamente, só falta suavizar os sopés das montanhas (região 4 das Figura 1.3, 1.4 e 1.5). Para isso, é necessário separar a região de interesse. Primeiramente, define-se novamente um limiar que representa qual a fração de área dos penhascos que será suavizada, a fim de se ter o relevo desejado. Esse é outro parâmetro que o usuário informa para o algoritmo e, com isso, mais uma vez, ele tem maior controle sobre as características do resultado gerado.

A abordagem, então, é selecionar a área a ser suavizada através da aplicação de um filtro espacial que irá suavizar as transições entre os penhascos dessa região. Essa suavização pode ser realizada através de um filtro de médias ou gaussiano. Basicamente, esses filtros trabalham com a influência dos pixels vizinhos e, dessa forma, é importante que as regiões que não fazem parte da região de interesse tenham pouco impacto no restante da suavização. Para isso ocorrer, é necessário que essas regiões apresentem uma intensidade

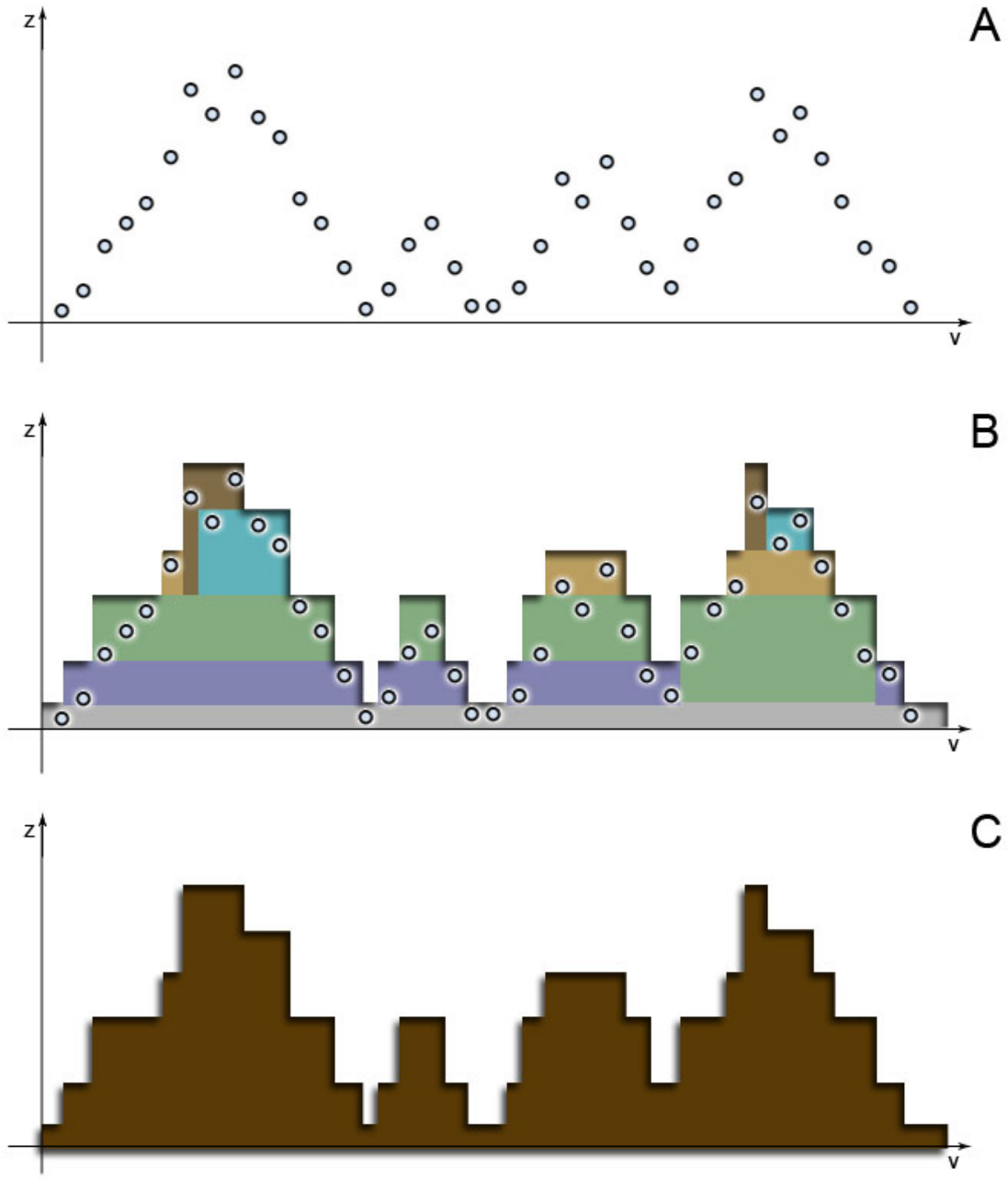

Figura 3.4: Aplicação do *Mean Shift* no espaço de características, sendo z, a altura, e v um vetor unidimensional contendo todos os pixels da imagem. Em (A), observa-se amostra inicial; em (B), observa-se a amostra e clusters definidos sobrepostos; por fim, em (C), apenas os clusters.

que se aproxime do valor do limiar usado. A equação 3.3 representa a função de limiar utilizada nesta dissertação, a fim de gerar o resultado mencionado. O passo seguinte é operar sobre a imagem da base um dos filtros espaciais mencionados. Nessa situação, é utilizado um *kernel* relativamente grande para criar uma transição suave. Para a definição da região do topo basta executar a operação de limiar de forma similar, como apresentada na equação 3.4.

$$base(x,y) = \begin{cases} img(x,y) & \text{se img(x,y)} < limiar \\ limiar & \text{caso contrário} \end{cases} \quad (3.3)$$

$$topo(x,y) = \begin{cases} img(x,y) - limiar & \text{se img(x,y)} > limiar \\ 0 & \text{caso contrário} \end{cases} \quad (3.4)$$

Basicamente as operações de limar para separa as região a ser suavizada (base) e o topo é igual. Contudo, o a região do topo apresenta a subtração pelo próprio limiar. Isso é necessário para a cominação da base suavizada e do topo. Compensa-se a intensidade do limiar na imagem do topo a fim de poder representar o mapa de alturas de maneira correta. O passo final é somar a imagem da base suavizada com o topo, gerando, assim, o cânion final.

3.2 Implementação

Objetivando uma melhor explicação sobre a implementação, o algoritmo 3.2.1 apresenta todos os passos utilizados para a codificação da solução proposta nesta dissertação. Para isso, o algoritmo em questão requer uma série de informações que são necessárias para a geração de resultados dos cenários 3D de cânions com grande variabilidade. Dentre as informações, o algoritmo de Perlin toma por parâmetros $pSeed$ (semente), a frequência e o número de oitavas para a geração do ruído, o que causa grande impacto na formação do relevo final do cânion, como apresentado na seção 4.3.1.

As variáveis w e h correspondem ao tamanho do mapa de altura, sendo w a largura e h a altura. Outro parâmetro que tem grande influência sobre o resultado final gerado é o raio – definido pela variável $mRaio$ – que é usado pelo algoritmo *Mean Shift*. A questão do impacto proveniente da escolha do raio do *Mean Shift* é apresentado e discutido no capítulo de resultados.

Seguindo as parametrizações necessárias, existe a fração das montanhas que serão suavizadas – variável $encostaFrac$ – e a fração de áreas de planícies – variável $planFrac$

– que representam as características 4 e 5 apresentadas no diagrama da Figura 1.3, respectivamente. A forma com que essas frações impactam no algoritmo serão descritas na sequência.

Para a definição do rio, existe apenas a necessidade de dois pontos P_1 e P_2, definidos pelo usuário, sendo que eles representam as coordenadas inicial e final do curso do rio.

A partir das parametrizações mencionadas, o algoritmo 3.2.1 gera um reticulado inicial, de dimensões w e h, usando o método do ruído de Perlin, que é abstraído pelo método $GerarRuidoDePerlin$.

O segundo passo identifica as regiões de planícies. Essa etapa está abstraída através do uso da função $Planicies$, que recebe como argumentos a imagem inicial gerada ($imgPerlin$) e o raio ($mRaio$) associado ao algoritmo *Mean Shift*. O resultado dessa função é uma estrutura de dados que encapsula a variável $imgPlaniciesBinaria$, que representa uma máscara binária contendo as planícies definidas pelos pixels de valor 1. O outro valor associado a essa estrutura é o $limiarInverso$, o qual foi utilizado previamente para a operação de *threshold*, que resultou na imagem $imgPlaniciesBinaria$. Toda a abordagem referente a definição das planícies é explicada na seção 3.3.

function CANION ($w, h, pOct, pSeed, pFreq, mRaio, encostaFrac, planFrac, P_1, P_2$)
▷ Tal que $planFract, encostaFrac \in \mathbb{R}$, onde que $0 \leq planFract \leq 1$ *e* $0 \leq encostaFrac \leq 1$
    $imgPerlin \leftarrow GerarRuidoDePerlin(w, h, pOct, pSeed, pFreq)$
    $(imgPlaniciesBinaria, limiarInverso) \leftarrow Planicies(ImgPerlin, mRaio,$
        $planFract)$ ▷ Definição das planícies

    $rotaAltura \leftarrow (L_m - limiarInverso)$ ▷ Sendo L_m o nível máximo de cinza
    $imgRotas \leftarrow ConexaoEntrePlanicies(imgPerlin, imgPlanciesBinaria,$
        $rotaAltura)$

    $imgPlaniciesERotas \leftarrow AplicarMascara(imgRotas, imgPlanciesBinaria,$
        $rotaAltura)$

    $imgPenhascosIntermediarios \leftarrow ExecMeanshift(imgPlaniciesERotas, mRaio)$
▷ Definição dos penhascos
    $listaDeClusters \leftarrow ListaOrdenadaDeIntensidadesClusters($
        $imgPenhascosIntermediarios)$

    $imgPercursoRioBinario \leftarrow ExecutaDijkstra(P_1, P_2, ImgPerlin)$ ▷ Definir curso do rio.

    $rioAltura \leftarrow \frac{rotaAltura}{2}$
    $imgCanionIntermediario \leftarrow AplicarMascara(imgPercursoRioBinario,$
        $imgPenhascosIntermediarios, rioAltura)$ ▷ Aplicar rio à imagem com os penhascos

    $tamLista \leftarrow tamanho(listaDeClusters)$
    $limiarTopo \leftarrow listaDeClusters[tamLista * encostaFrac]$
    $imgMascaraBinariaTopo \leftarrow threshold(imgCanionIntermediario,$
        $limiarTopo)$ ▷ Separa região de interesse das partes mais altas
    $imgMascaraBinariaImgBase \leftarrow negativo(imgMascaraBinariaTopo)$ ▷ Separa região de interesse das partes mais baixas

    $compensa \leftarrow \frac{listaDeClusters[0]}{2}$
    $imgPreSuavizacao \leftarrow PreSuavizacao(imgCanionIntermediario,$
        $imgMascaraBinariaImgBase, limiarTopo, compensa)$
    $tamKernel \leftarrow 15$
    $imgBase \leftarrow DesfocagemGaussiana(imgPreSuavizacao, tamKernel)$ ▷ Suavizar image das partes mais baixas
    $imgTop \leftarrow PreTopo(imgCanionIntermediario, imgMascaraBinariaTopo,$
        $limiarTopo)$ ▷ Prepara regiões mais altas para mesclar

    $imgCanion \leftarrow imgBase + imgTop$
    return $imgCanion$
end function

Algoritmo 3.2.1: Geração procedural de cânions.

Considerando-se o fato das planícies terem sido definidas a partir do negativo da imagem inicial, faz-se necessário converter esse valor para o contexto da imagem inicial. Essa conversão é definida pela variável $rotaAltura$, a qual define tanto a altura das planícies, quanto as rotas que as conectam. Dessa forma, calcula-se a variável $rotaAltura$ subtraindo-se o valor máximo representável (L_m) pelo limiar recebido pela função de definição de planícies ($limiarInverso$).

A definição das rotas é dada através da função $ConexaoEntrePlanicies$ que retorna uma imagem (Figura 3.5 B) contendo as rotas que conectam as planícies. Destaca-se o fato de que a imagem resultante das rotas é gerada a partir de uma cópia da imagem inicial, onde as rotas são desenhadas com a intensidade certa referente à altura. A definição detalhada da função $ConexaoEntrePlanicies$ é apresentada na seção 3.4.

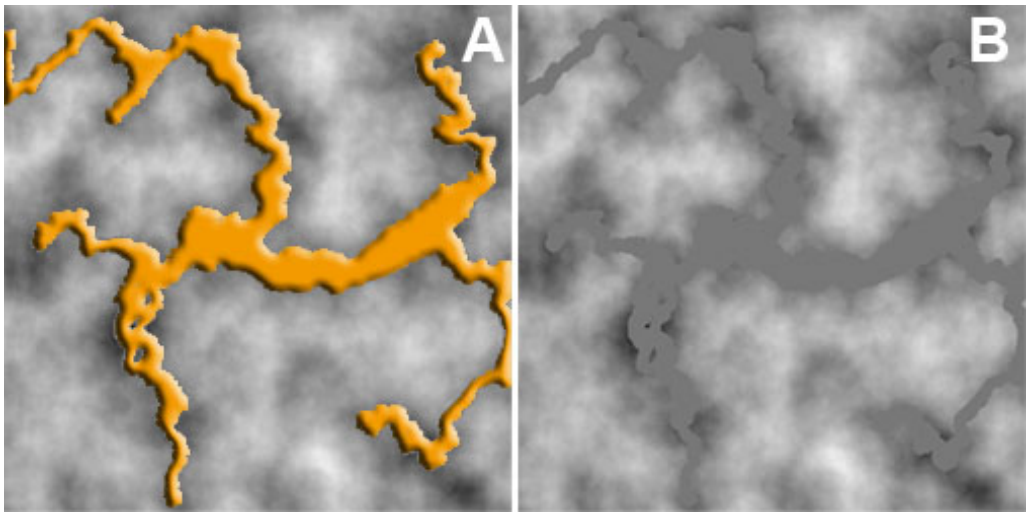

Figura 3.5: Rotas de conexão entre planícies, sendo em B o mapa gerado e em A destaque das rotas geradas conectando todas as planícies

O próximo passo é a aplicação das planícies na Figura 3.5 B. O resultado pode ser observado na Figura 3.6 A. Nesse monento tem-se as planícies e suas rotas de conexão geradas e sobrepostas na imagem inicial. Ao executar o algoritmo *Mean Shift* obtem-se um terreno repleto de penhascos como se fossem uma sucessão de degraus, como pode ser visto na Figura 3.6 B.

Na sequência, a rota do rio é definida a partir da aplicação do algoritmo de Dijkstra sobre a imagem inicial, tendo P_1 e P_2 (previamente definidos pelo usuário) como pontos de entrada. De posse da rota do rio, define-se a altura que o rio irá assumir no mapa de alturas, e aplica-se a máscara de sua rota sobre a imagem previamente clusterizada com

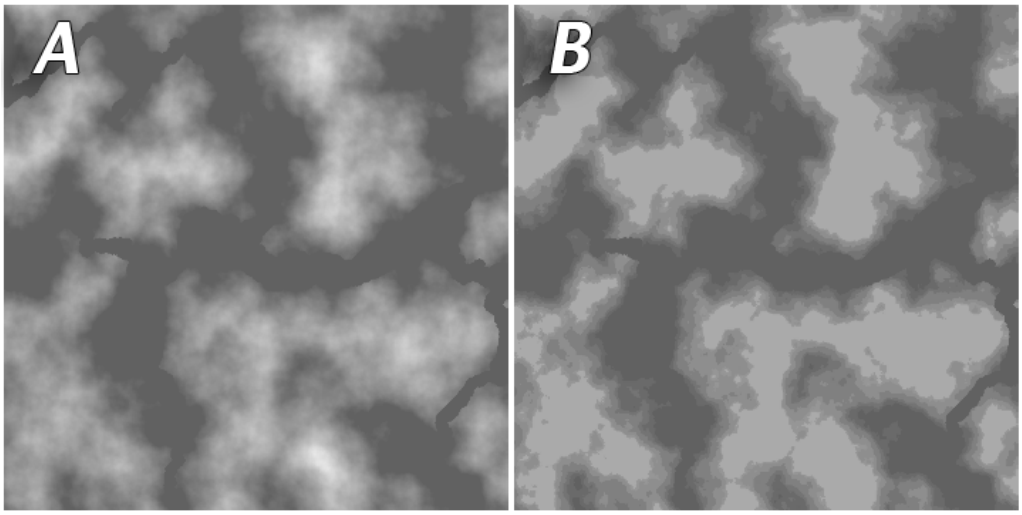

Figura 3.6: Aplicação do algoritmo *Mean Shift* sobre as planícies e suas conexões gerando os penhascos.

o algoritmo *Mean Shift*. A altura do rio é definida na linha 19 do algoritmo 3.2.1, na qual usa-se metade do valor associado à altura das planícies e suas conexões. O resultado gerado até o momento é o cânion intermediário que, ainda, não leva em conta o algoritmo de suavização do sopé das montanhas. Visualiza-se o cânion intermediário na Figura 3.7.

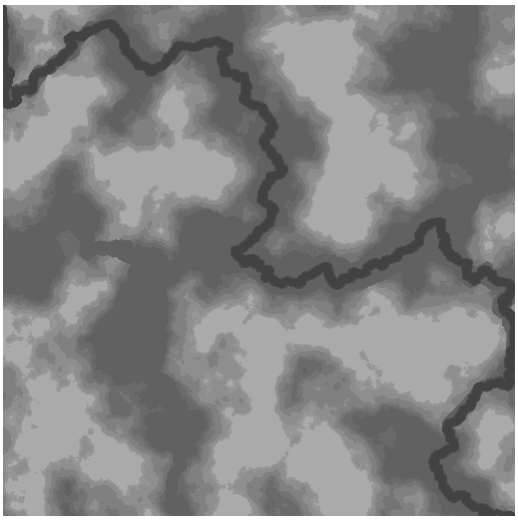

Figura 3.7: Aplicação do curso do rio sobre os penhascos (cânion intermediário).

Os próximos passos estão relacionados a definição das encostas suaves e, para isso, é necessário separar a região de interesse que se pretende suavizar. Dessa forma, a estratégia adotada baseia-se em definir duas máscaras binárias, uma para a região que será

definida por penhascos, e outra que será suavizada. Assim sendo, é necessário definir uma operação de *threshold* que separem essas regiões, sendo que, de posse da máscara da região mais alta, executa-se uma operação de negativo para se ter as regiões mais baixas.

Para isso, é importante contextualizar que a definição dos penhascos, feita a partir da operação de *Mean Shift*, resulta, além da imagem segmentada, em um vetor ordenado de intensidades. Exemplificando, caso existam 3 clusters na imagem segmentada, de intesidades 56, 201 e 97, respectivamente, tem-se o vetor ordenado $[56, 97, 201]$. Um processo semelhante é realizado na definição das planícies, como é apresentado na seção 3.3.

O algoritmo irá definir a intensidade de nível de cinza utilizada na operação de *threshold* a partir do vetor ordenado, mencionado anteriormente. Dessa forma, o valor do limiar usado para o *threshold* baseia-se na escolha do índice que irá servir de base para a operação, e isso é definido através da variável $encostaFrac$. Em outras palavras, caso a fração de encosta suave seja 0.4 (40%) em um vetor contendo 10 elementos, de índices de 0 a 9, opta-se pela intensidade do elemento de índice 4. A Figura 3.8 B exemplifica a máscara binária das regiões de penhasco e planaltos, sendo C a região em questão selecionada da imagem A.

Figura 3.8: Separação das regiões de interesse das partes mais altas.

O passo seguinte é separar a região a ser suavizada e aplicar uma técnica de interpolação, neste caso, o borramento Gaussiano, para definir as encostas suaves. A região de interesse a ser suavizada é definida pela equação 3.3.

Outra questão relevante é o fato da aplicação do algoritmo *Mean Shift* influenciar a escala de representação dos dados. Ou seja, o mapa gerado inicialmente pela técnica de Perlin apresenta a menor intensidade de um pixel pelo valor 0, e a aplicação da segmen-

tação resultará em um valor $x > 0$. Dessa forma, a compensação desse valor pode ser interessante para facilitar a utilização do mapa de altura criado. A implementação adotada nesta dissertação (algoritmo 3.2.2) apresenta a implementação da equação 4.1, a qual leva em conta a compensação mencionada e a escolha da região de interesse, de forma a se ter um resultado satisfatório com a desfocagem Gaussiana. A Figura 3.9 apresenta em A o cânion intermediário e em B a região a ser suavizada, na parte cinza mais escura.

$$destino(x,y) = \begin{cases} img(x,y) - compensa & se\ img(x,y) < limiar \\ limiar - compensa & \text{caso contrário} \end{cases} \quad (3.5)$$

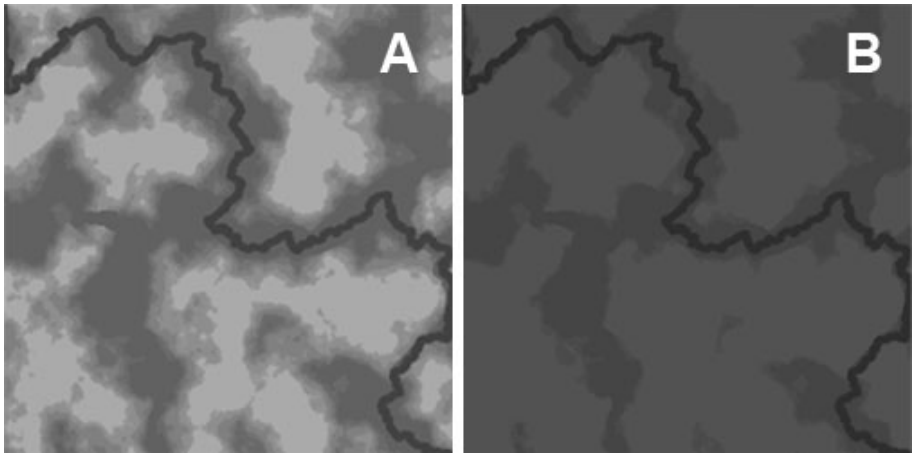

Figura 3.9: Separação da região de interesse das partes mais baixas.

1: **function** PRESUAVIZACAO($imgPreCanion, imgBinBase, limiar, compensa$)
2: $tam \leftarrow (largura(imgPreCanion) \times altura(imgPreCanion))$
3: $imgPreSuavizacao \leftarrow criarImagem(tam)$
4: **for** $i \leftarrow 0, tam$ **do**
5: **if** $imgBinBase[i] = 1$ **then**
6: $imgPreSuavizacao[i] \leftarrow imgPreCanion[i]$
7: **else**
8: $imgPreSuavizacao[i] \leftarrow limiar$
9: **end if**
10: **end for**
11: $imgPreSuavizacao \leftarrow SubEscalar(imgPreSuaviacao, compensa)$
12: **return** $imgPreSuavizacao$
13: **end function**

Algoritmo 3.2.2: Preparação da imagem base para suavização.

Uma vez realizada a pré-suavização, aplica-se o borramento Gaussino, usando um kernel grande (15 x 15). Na sequência, prepara-se a região de interesse do topo (Fi-

gura 3.8 C) para somar com a região de encostas suaves, subtraindo todos os pixels pelo limiar usado pela sua máscara. Dessa forma, mantem-se uma altitude similar a que se tinha. Por fim, como visto na Figura 3.10, soma-se a área de encostas suaves (A) com o topo (B), gerando o mapa de altura do cenário do cânion (C).

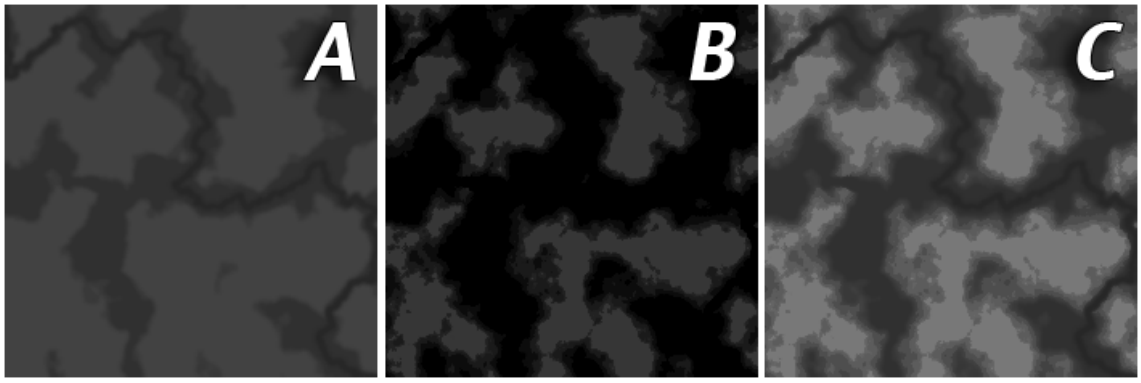

Figura 3.10: Combinação da região de encostas suaves (A) com os penhascos (B) gerando o cânion final (C).

A Figura 3.11 apresenta um resumo dos passos executados no algoritmo. As etapas são listadas na sequência:

- (1) Geração do ruído de Perlin;
- (2,3,4) Definição das planícies (algoritmo *Mean Shift*);
- (5,6) Definição das rotas entre as planícies (rotulagem de componentes conectados de CHANG et al. (2004); algoritmo de Dijkstra);
- (7) Aplicação da máscara das planícies;
- (8) Definição dos penhascos (algoritmo *Mean Shift*);
- (9) Definição do curso do rio (algoritmo de Dijkstra);
- (10) Cânion intermediário;
- (11) Definição da máscara regiões altas (topo);
- (12) Definição da máscara regiões baixas – sopé da montanha;
- (13,15,17) Preparação das regiões baixas para suavização;

- (14,16) Preparação das regiões altas para gerar cânion final;
- (19) Cânion final;

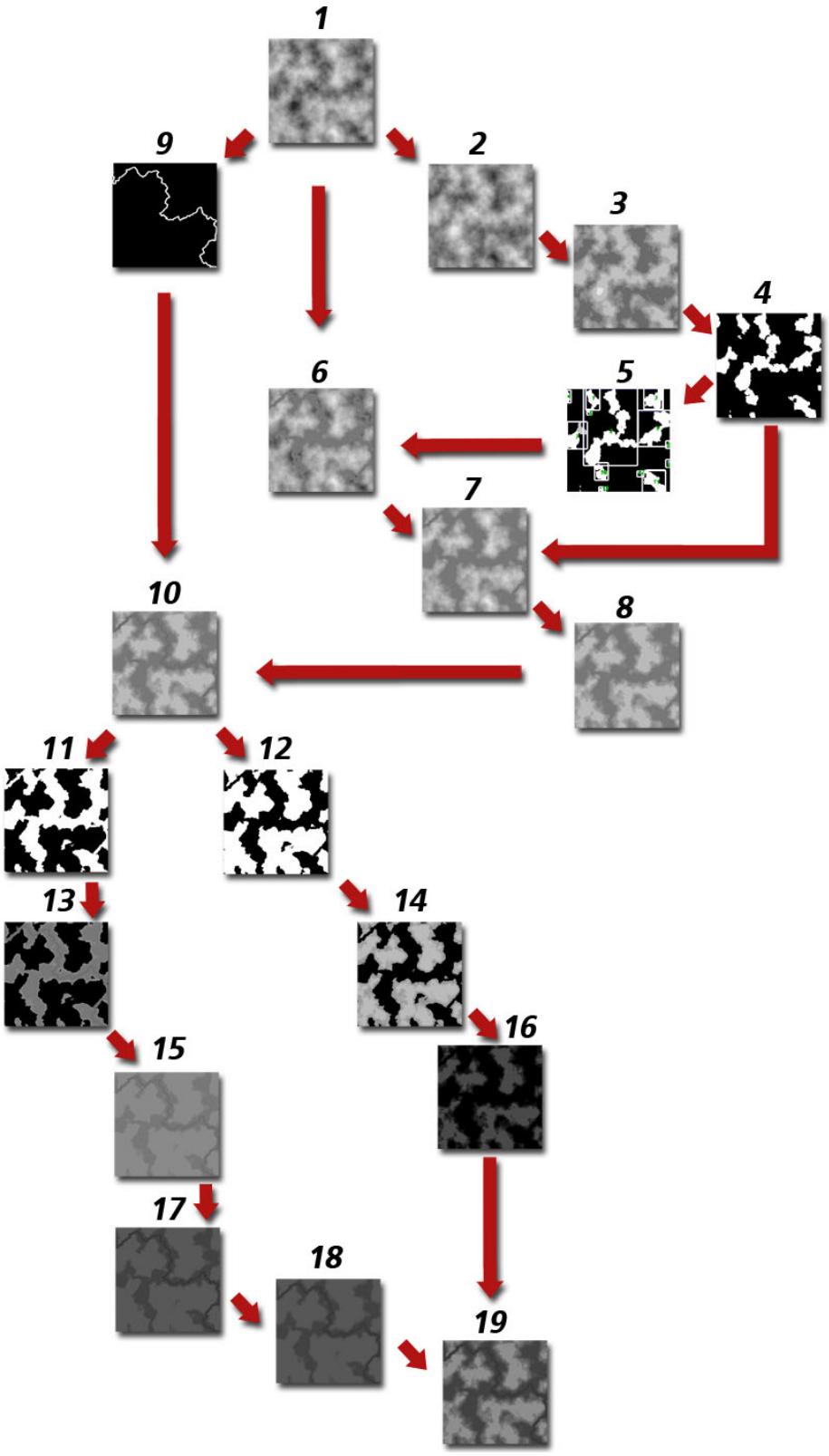

Figura 3.11: Visão geral da implementação do algoritmo de geração de Cânions.

3.3 Definição das planícies

O processo completo que define as regiões de planícies é executado no algoritmo 3.3.1, e representa as regiões de números 2, 3 e 4 do diagrama da Figura 3.11. Optou-se por identificar as regiões de maior altitude na imagem negativa do ruído de Perlin (região 2 da Figura 3.11) por considerar mais intuitivo a operação de *Threshold* sobre as regiões mais altas, conforme equação 3.3.

$$destino(x,y) = \begin{cases} 1 & se\ img(x,y)\ > T(x,y) \\ 0 & \text{caso contrário} \end{cases} \tag{3.6}$$

1: **function** PLANÍCIES ($imgPerlin, mRaio, planFract$)
2: $imgPerlinNegativo \leftarrow negativo(imgPerlin)$
3:
4: $imgClust \leftarrow ExecMeanshift(imgPerlinNegativo, mRaio)$
5: $listaDeIntensidades \leftarrow ListaOrdenadaDeIntensidadesClusters(imgClust)$
6: $tamLista \leftarrow tamanho(listaDeIntensidades)$
7:
8: $indice \leftarrow tamLista * planFract$ ▷ $indice \in \mathbb{Z}$, *onde que* $0 \leq indice$ e $planFract \in \mathbb{R}$, *onde que* $0 \leq planFract \leq 1$
9:
10: $limiar \leftarrow listaDeIntensidades[indice]$
11:
12: $imgRegPlanicies Binaria \leftarrow threshold(imgPerlinNegativo, limiar)$
13: **return** $(imgRegPlanicies Binaria, limiar)$
14: **end function**

Algoritmo 3.3.1: Definição das regiões de planícies.

O resultado obtido na etapa de definição das regiões é uma imagem binária que corresponde às regiões de planícies. A aplicação do algoritmo *Mean Shift* com um raio $mRaio$ sobre a imagem negativa gera uma formação que lembra um cânion, sendo que as regiões mais altas correspondem às regiões baixas da imagem original. Além da imagem segmentada, tem-se um vetor ordenado contendo a intensidade referente a cada cluster, gerado de forma semelhante ao vetor usado na definição dos penhascos, apresentado anteriormente. Esse vetor é um subproduto da segmentação usada, e a sua obtenção é abstraída pela função $ListaOrdenadaDeIntensidadesClusters(imgClust)$.

Dessa forma, seleciona-se o valor do cluster desejado no vetor e faz-se a operação de $threshod$ sobre a imagem negativa, tendo, assim, uma imagem binária contendo as planícies. O valor sobre o qual o $threshod$ irá operar baseia-se na escolha de um elemento

do vetor de intensidades. A escolha do elemento é feita através de seu índice. Para isso multiplica-se a variável $planFract$ pelo número de elementos do vetor, o que resulta no valor usado para o limiar. Em outras palavras, se a fração $planFract$ for 0,3 (30%), por exemplo, e o vetor de clusters tiver 10 elementos, tem-se o índice 3. O resultado gerado é visto na Figura 3.11 na imagem 4.

3.4 Definição do curso do rio e conexão entre as planícies

Tanto as conexões entre as planícies quanto a definição do curso do rio seguem a mesma abordagem. O princípio da mínima ação explica que a natureza é econômica em todas as suas ações. Em outras palavras, uma partícula tende a executar a ação que irá lhe despender menor quantidade de energia. Exemplifica-se, a formação de um rio real não leva em consideração o menor caminho, mas sim o caminho que a água enfrente a menor resistência. E é por isso que encontra-se na natureza rios com muitas variações em seu curso e outros com curso menos acidentados.

Dessa forma, procurou-se desenvolver uma lógica que esteja de acordo com o princípio da mínima ação. Isso é feito através de uma função que analisa o declive do terreno de forma a tender sempre para a região mais baixa. O algoritmo usa o reticulado criado a partir do ruído de Perlin como entrada para o cálculo do custo aplicado ao algoritmo de Dijkstra, o qual define todas as rotas.

O custo do deslocamento de um ponto P_1 (inicial) para um ponto P_2 (final) leva em conta a diferença de altura, sendo que ele baseia-se no somatório da transição do custo entre os vértices vizinhos do reticulado (equação 3.8), e isso é feito ao subtrair a altura de um ponto pela altura do próximo ponto (equação 3.7). A descrição completa do custo associado a rota é apresentada na sequência.

Um ponto a destacar é que essa abordagem pode resultar em custos negativos. O algoritmo de Dijkstra, contudo, não foi projetado para trabalhar sobre custos de arestas menores ou iguais a zero, o que em primeira análise parece ser um impedimento. Entretanto, apesar do custo de transição entre cada vértice poder assumir valores negativos, a abordagem proposta apresentou resultados satisfatórios para todos os testes, como exemplificado na Figura 3.12. Acredita-se que isso seja resultado da maneira que o ruído de Perlin é gerado, o qual dificilmente irá criar um relevo completamente plano ou que tenha declive constante para uma mesma direção. Durante esses testes, a função de custo conse-

guiu gerar formações complexas, tendendo sempre para a posição mais baixa do terreno, o que lembra muito a natureza.

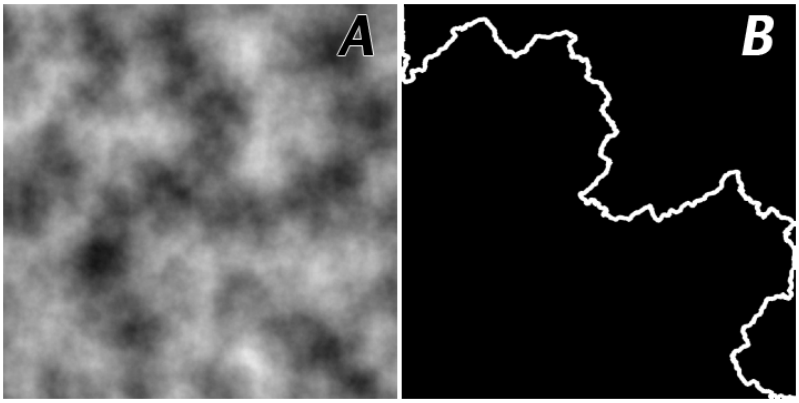

Figura 3.12: Definição do curso do rio com o algoritmo de Dijkstra, sendo (A) o reticulado inicial definido pelo ruído de Perlin e (B) a rota resultante.

Formalmente, considera-se uma região $\Omega \in R^2$ e dois pontos (P_1 e P_2), na qual objetiva-se calcular um caminho contínuo ρ de P_1 até P_2 que minimiza uma função de custo ao longo do caminho. O custo da transição entre dois vértices vizinhos é definido pela equação 3.7, onde $z(p)$ corresponde a altura do terreno para o vértice em questão, sendo a vizinhança 8 conectada. Define-se $\mathcal{P}$ como sendo o conjunto de todos os caminhos contínuos em Ω de P_1 à P_2, ou seja, $\mathcal{P}$ denota o conjunto de todas as funções de custo contínuas $\rho : [0, T] \rightarrow \Omega, T > 0$, onde $\rho(0) = P_1$ e $\rho(T) = P_2$. Considera-se $C : \mathcal{P} \rightarrow [0, \infty($ onde $\rho \in \mathcal{P}$ denotando o custo de um caminho $\rho \in \mathcal{P}$ na equação 3.8.

$$k(p_i, p_{ii}) = z(p_{ii}) - z(p_i) \tag{3.7}$$

$$C(\rho) = \sum_{i=0}^{T-1} k(p_i, p_{i+1}) \tag{3.8}$$

O caminho desejado é um caminho $\rho*$ que minimiza a função de custo $C(\rho)$:

$$C(\rho*) = \min_{\rho \in \mathcal{P}} C(\rho) \tag{3.9}$$

Para cada rota gerada – seja ela curso do rio ou a conexão entre duas planície – tem-se uma máscara binária onde o conjunto dos pixels de valor 1 representa a rota, como é visto na Figura 3.12B. No caso das rotas entre as planícies, a lógica aplicada atua conforme o algoritmo 3.4.1. Ou seja, tendo por entrada uma imagem binária contendo as regiões

de planícies, aplica-se o algoritmo de rotulagem de componentes conectados de CHANG et al. (2004) (Figura 3.13A), e tem-se como resultado, além da rotulagem em si, um conjunto de centróides resultantes. O próximo passo é fazer uma cópia da imagem do ruído de Perlin, com objetivo de não fazer nenhuma alteração no reticulado incial. Essa cópia em questão recebe o nome de $imgConjuntoDeRotas$ (linha 5 algoritmo 3.4.1) e apresenta exatamente a mesma informação da imagem gerada pelo ruído de Perlin.

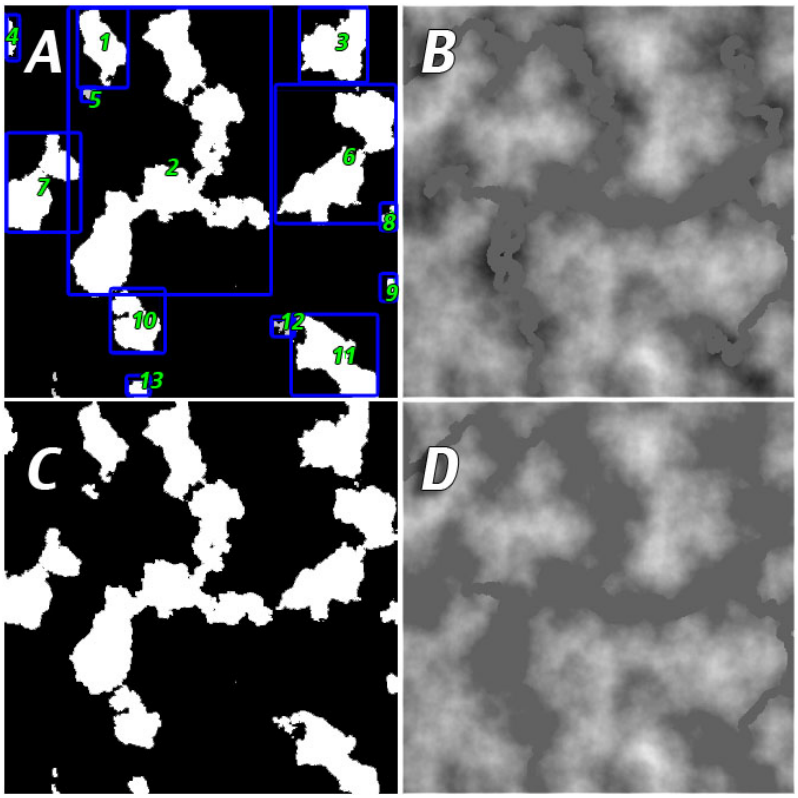

Figura 3.13: Aplicação das rotas e planícies sobre o ruído de Perlin inicial.

```
1: function CONEXAOENTREPLANICIES(imgRotas, imgRegPlaniciesBin, intensidade)
2:     imgPlaniciesRotuladas ← ExecutaRotulagemDeComponentesConectados(
3:                                 imgRegPlanciesBin)
4:     listaDeCentroides ← ListaDeCentroides(imgPlaniciesRotuladas)
5:     imgConjuntoDeRotas ← imgBase
6:     for all centroide ∈ listaDeCentroides do
7:         if centroide ≠ ultimoCentroide ∈ listaDeCentroides then
8:             centroide2 ← proximoCentroide ∈ listaDeCentroides
9:             imgRotaBinaria ← ExecutaDijkstra(centroide, centroide2,
10:                                             imgConjuntoDeRotas)
11:            imgConjuntoDeRotas ← AplicarMascara(imgConjuntoDeRotas,
12:                                        imgRotaBinaria, intensidade)
13:        end if
14:    end for
15:    return imgConjuntoDeRotas
16: end function
```

Algoritmo 3.4.1: Definição das conexões entre as planícies.

Com relação aos conjunto dos centróides encontrados na etapa anterior, deseja-se ter uma árvore – não sendo necessariamente uma árvore geradora mínima (CORMEN et al., 1999) – que conecte todos centróides. O algoritmo 3.4.1 apresenta a abordagem implementada considerando-se sucessivas iteraçãos sobre a imagem $imgConjuntoDeRotas$. Ao término do processo tem-se a imagem resultante na Figura 3.13B. A figura em questão também apresenta a aplicação da máscara binária das planícies (C) sobre a imagem resultante da aplicação das rotas (B), gerando a imagem (D), que será usada como entrada para definição dos penhascos.

Dessa forma, é proposto como trabalho futuro a elaboração de um algoritmo para a criação de rotas alternativas para permitir uma maior interligação entre as áreas de planície, de forma que se possa chegar ao mesmo centróide por mais de um caminho.

Já no caso do rio, após a etapa de definição dos penhascos, aplicada-se a máscara do curso do rio sobre a imagem que contem os penhascos (Figura 3.6 B), sendo que a altura do rio é definida pela metade da altura utilizada para a criação das planícies, como pode ser visto na linha 19 do algoritmo 3.2.1. A Figura 3.7, por fim, apresenta o resultado da aplicação do curso do rio sobre a imagem contendo os penhascos. Dessa forma, tem-se o cânion intermediário que é apresentado no diagrama UML de atividades da Figura 3.2.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos nesta dissertação. Para facilitar a comparação dos resultados com os objetivos propostos, optou-se por ilustrar imagens de cânions reais que se deseja imitar (seção 4.1), bem como as características que não se deseja imitar (seção 4.2).

A seção 4.3 apresenta os resultados de forma visual sobre o prisma das considerações de realismo, escala, variações e controle apresentadas por GEORGE e HUGH (2006). Por fim, a seção 4.4 apresenta testes de desempenho para a geração do mapa de altura desejado, sendo que, nessa última seção, optou-se pelo uso do termo "teste de carga", ao invés de "desempenho", para evitar uma possível confusão sobre o desempenho associado à jogabilidade.

4.1 Cânions Reais: Características desejadas

Dentre as imagens pesquisadas, as Figuras 4.1 a 4.8 ilustram os parâmetros visuais para a avaliação das características presentes em cânions. Para a pesquisa, realizou-se a busca por imagens de cânions no repositório *online Flickr* (http://www.flickr.com/) e na enciclopédia colaborativa *Wikipedia*, usando as palavras chave “Cânion” e “Canyon”. Como condição, as imagens deveriam ser de domínio público ou com alguma licença do tipo *Creative Commons*.

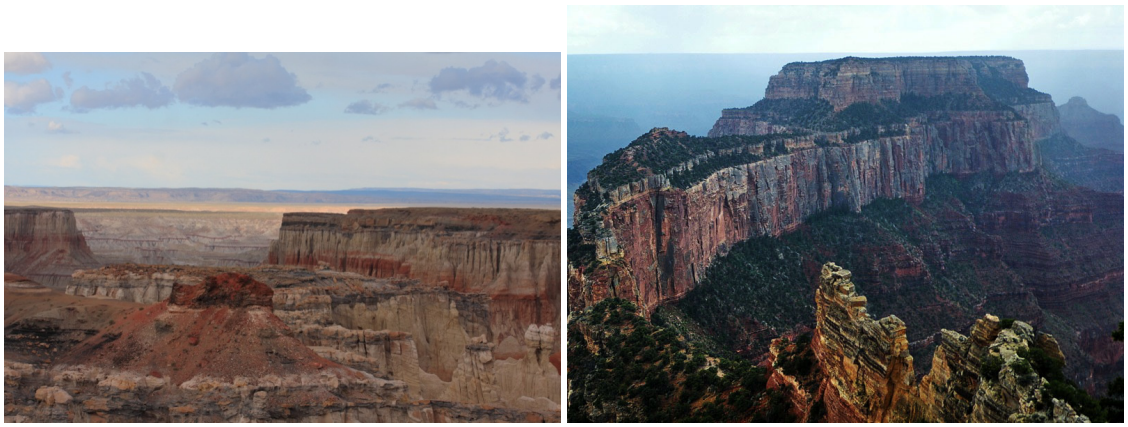

Figura 4.1: *Coal Mine Canyon, Arizona* (COBALT123, 2010), *Grand Canyon North Rim, Arizona, US* (AL_HIKESAZ, 2009).

Figura 4.2: *Chiricahua Mountains, Cave Creek Canyon, Portal, Arizona, US* (BALVARIUS, 2009), *Bryce Canyon, Utah, US* (SIMMER, 2004).

Figura 4.3: *Canyon de Chelly National Monument, Arizona, USA* (NEGAUNEE, 2009), *Canyon de Chelly NM AZ.* (WELLER e WELLER, 2005).

Figura 4.4: *Horseshoe Bend, Glen Canyon Dam, Arizona* (MEHLFÜHRER, 2008), *Grand Canyon* (THIERRY, 2010)

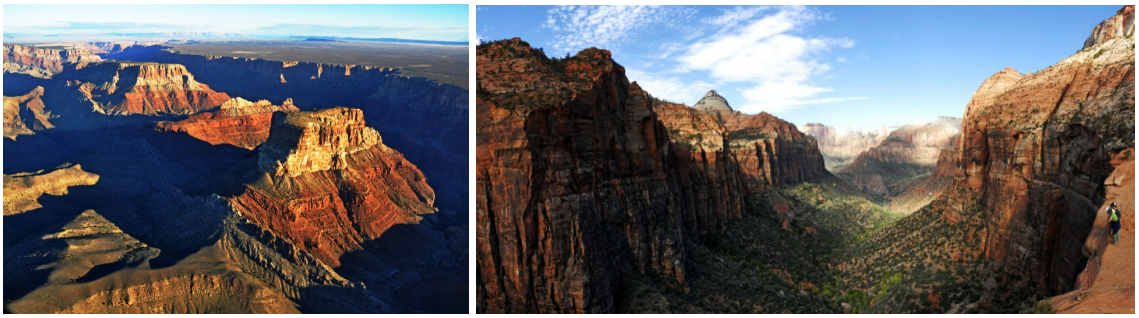

Figura 4.5: *Grand Canyon DEIS Aerial: Kwagunt Butte, Malgosa Crest, Nankoweap Mesa,Arizona, US* (SERVICE, 2010), *Canyon Overlook II, Utah, US.* (LAU, 2007b).

Figura 4.6: *Spider Rock, Apache County, Arizona, US* (HEY, 2009b), *Glen Canyon* (PHELPS, 2004).

Figura 4.7: *Grand Canyon Helicopter Flight (20), Arizona, US* (MACDONALD, 2009), *First canyon, Cameron, Arizona, US.* (PAPHIO, 2006).

Figura 4.8: *Grand Canyon west* (KANNAN, 2008), *Santa Elena HDR, Brewster County, Texas, US* (MOON, 2006).

4.2 Cânions Reais: Características fora do escopo

A fim de ilustrar o que não se objetiva replicar, as Figuras 4.9 e 4.10 apresentam a visão aérea do *Grand Canyon*. Observa-se que todo o relevo tende a seguir o curso do rio, sendo isso uma das características que não se buscou replicar. Para essas figuras, buscou-se pesquisar as imagens de satélite no *software* Google *Earth* (GOOGLE, 2011).

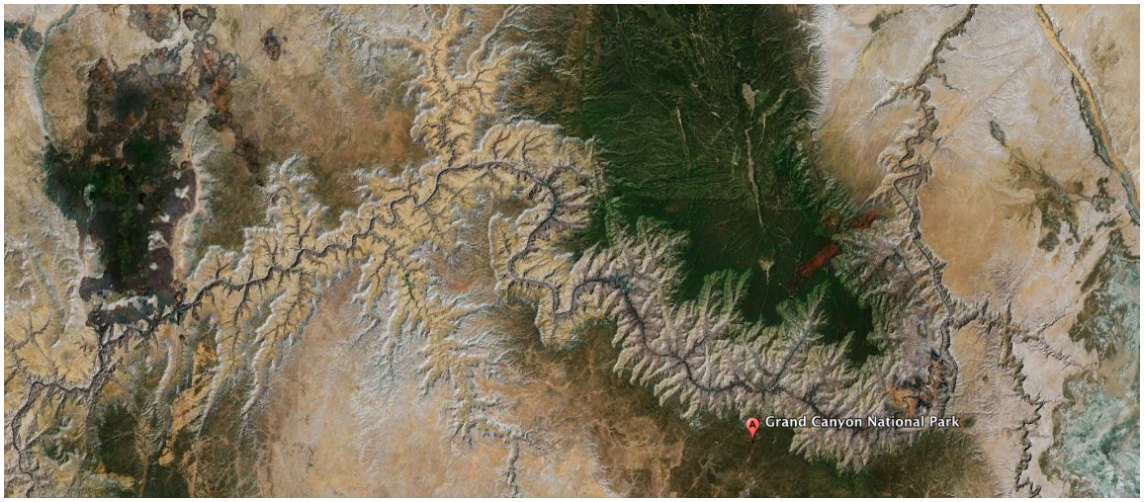

Figura 4.9: *Grand Canyon* visão aérea 1 (GOOGLE, 2011).

Figura 4.10: *Grand Canyon* visão aérea 2 (GOOGLE, 2011).

4.3 Cânions Procedurais

A grande maioria dos trabalhos de geração procedural na área de terrenos utiliza ferramentas gráficas consagradas para renderização dos resultados, mecanismo que permite uma melhor visualização das técnicas em situações práticas. Dessa forma, assim como no trabalho de ZHOU et al. (2007), utilizou-se o *software* Terragen (PLANETSIDE, 2011) para renderizar e apresentar os resultados.

Nas Figuras de 4.1 a 4.8, nota-se a influência das cores e formas impressas nas paredes dos cânions e isso tem grande impacto sobre o realismo, sendo uma característica desejável. Contudo, esta característica está fora do escopo desta dissertação. Dessa forma, a criação de texturas que imitem esse aspecto é sugerida como trabalho futuro. Outra característica evidenciada é a vegetação e sua distribuição, que também está fora dos objetivos e, da mesma forma, é sugerida como trabalho futuro.

As Figuras 4.9 e 4.10 apresentam a visão aérea do *Grand Canyon*. Apesar de todo o relevo tender a seguir o curso do rio, sendo essa uma das questões fora do escopo, é possível observar o curso do rio Colorado, no qual a rota é bastante complexa e acidentada. Essa última característica é atingida com o algoritmo desenvolvido nesta dissertação, gerando, assim, o curso do rio acidentado e complexo, tal qual o do rio Colorado.

Como apresentado na seção 1.1, George e Hugh (2006) enumeram uma série de quesitos que devem nortear a geração procedural, que são: realismo, escala, variações e controle. Dessas forma, os resultados serão discutidos conforme os quesitos mencionados.

4.3.1 Controle

Como apresentado no capítulo 3, o algoritmo necessita de diversos parâmetros de entrada para a geração do terreno. Todas essas entradas influenciam de alguma forma na síntese do terreno e são tratadas de maneira individual. Primeiramente, trata-se a questão da escolha da frequência do ruído de Perlin no resultado atingido. A Figura 4.11 apresenta a influência da escolha da frequência. Nota-se nessa imagem item A, que em frequências mais altas, tem-se o cânion com maior quantidade de regiões de planícies, sendo identificados 50 centróides de planícies. O mapa de altura gerado no item B apresenta menor quantidade de centróides (20), mas preserva as características de um cânion real. Por fim, é possível notar no item C que frequências muito baixas (apenas 2 centróides) não geram o resultado parecido com cânions, visto que a formação resultante somente lembra uma

escadaria.

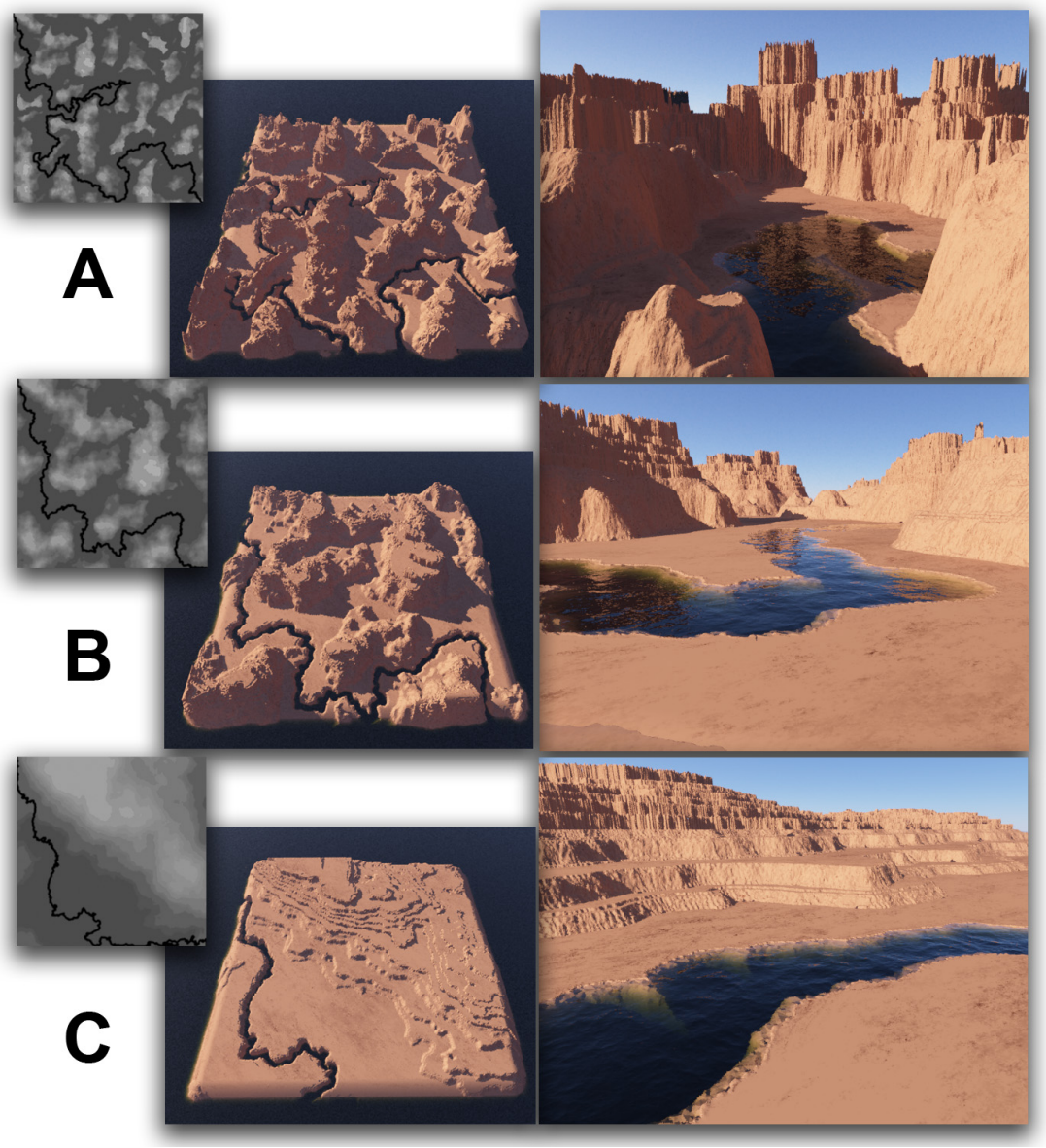

Figura 4.11: Influência da parametrização da frequência do ruído de Perlin no relevo gerado. Todas as imagens foram geradas com os mesmos parâmetros (arbitrários), exceto a frequência. A semente para o ruído de Perlin, o número de oitavas e o raio usado no *Mean Shift* são 93, 16 e 6, respectivamente. Já as frequências são 0,009 para (A), 0,005 para (B) e 0,001 para (C).

Para valores de frequência muito acima de 0,009 (como 0,02), o algoritmo torna-se muito lento pela quantidade de rotas necessárias a serem calculadas. Além disso, o resultado gerado não se assemelha aos cânions desejados, pois os dados ficam muito fragmentados e dispersos.

Como apresentado na revisão de literatura (seção 2.3), o ruído de Perlin usa o con-

ceito de oitavas, que define o uso de frequências sucessivas onde cada uma tem o dobro da frequência anterior. Dessa forma, o número de oitavas é outro fator que influencia nos resultados. A Figura 4.12 apresenta a influência do número de oitavas do ruído de Perlin na geração dos cenários de cânions. De maneira geral, o uso de 8 ou 16 oitavas gera resultados muito próximos e bons. O resultado do uso de 4 oitavas apresenta uma quantidade menor de informações, o que gera o terreno com regiões mais curvas e com menor complexidade visual. Por fim, o uso de duas oitavas foi capaz de gerar um terreno que em nada lembra a formação de um cânion.

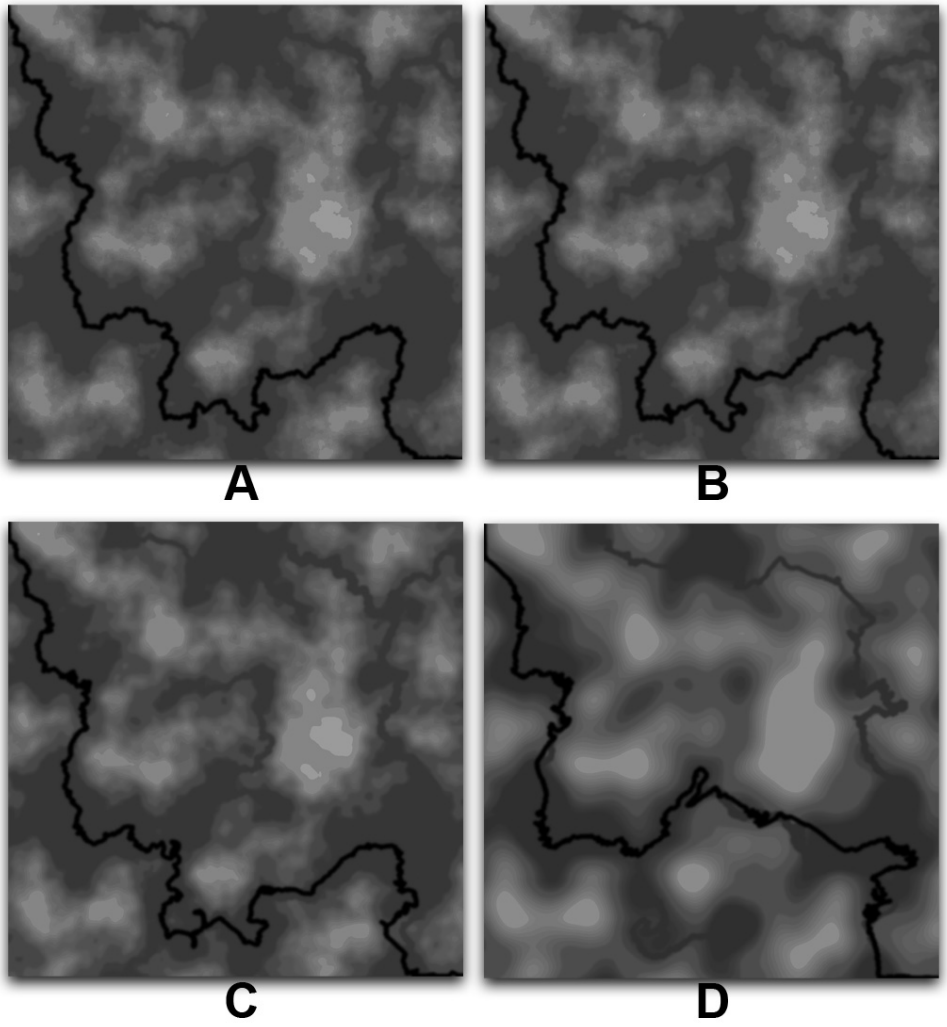

Figura 4.12: Influência do número de oitavas do ruído de Perlin para a formação do Cânion. A figura apresenta 4 mapas de altura, sendo (A) definido por 16 oitavas; (B), 8 oitavas; (C), 4 oitavas e (D), duas oitavas

O próximo item a ser discutido refere-se a influência da escolha do raio passado ao algoritmo *Mean Shift*. A Figura 4.13 apresenta essa questão sob a mesma entrada, variando-

se apenas o raio do *Mean Shift*. A Tabela 4.1 apresenta os parâmetros do raio usado, bem como resultados quantitativos para a mesma entrada. Nesse caso, utilizou-se para a semente um mesmo número arbitrário, bem como 0,3 (30%) de planícies e 0,3 (30%) de declive suave.

É possível observar que o terreno apresenta mais área de planície e um menor número de penhascos, na medida em que se aumenta o raio de entrada do *Mean Shift*, sendo a linha **A** o ruído de Perlin sem nenhum processamento. Na Figura 4.13, a coluna 1 mostra o mapa de altura; na coluna 2 apresenta-se a visão aérea da renderização; e, por fim, na coluna 3 apresenta-se a visão de um personagem imerso no cenário do cânion em questão. Durante a avaliação de diversos testes, evidenciou-se que os resultados mais parecidos com os cânions identificados durante os testes possuíam o raio variando entre 4 e 9. Contudo, essa é uma questão subjetiva, além de ter influência de outras parametrizações do ruído de Perlin, passível de interpretações visuais distintas, devendo o usuário definir o melhor valor para o seu caso.

Outro elemento que influencia o cânion gerado é a *seed* (semente) recebida pelo algoritmo de Perlin. Contudo, essa questão será tratada nas seções 4.3.2 e 4.3.3.

Tabela 4.1: Influência da escolha do raio do *Mean Shift* (Parâmetros para gerar a Figura 4.13)

	Raio (*Mean Shift 1 e 2*)	Clusters (*Mean Shift 1*	*Mean Shift 2*)	Centróides
A	-	-	-	-
B	3	56	40	27
C	4	38	26	24
D	5	25	18	26
E	9	8	7	12

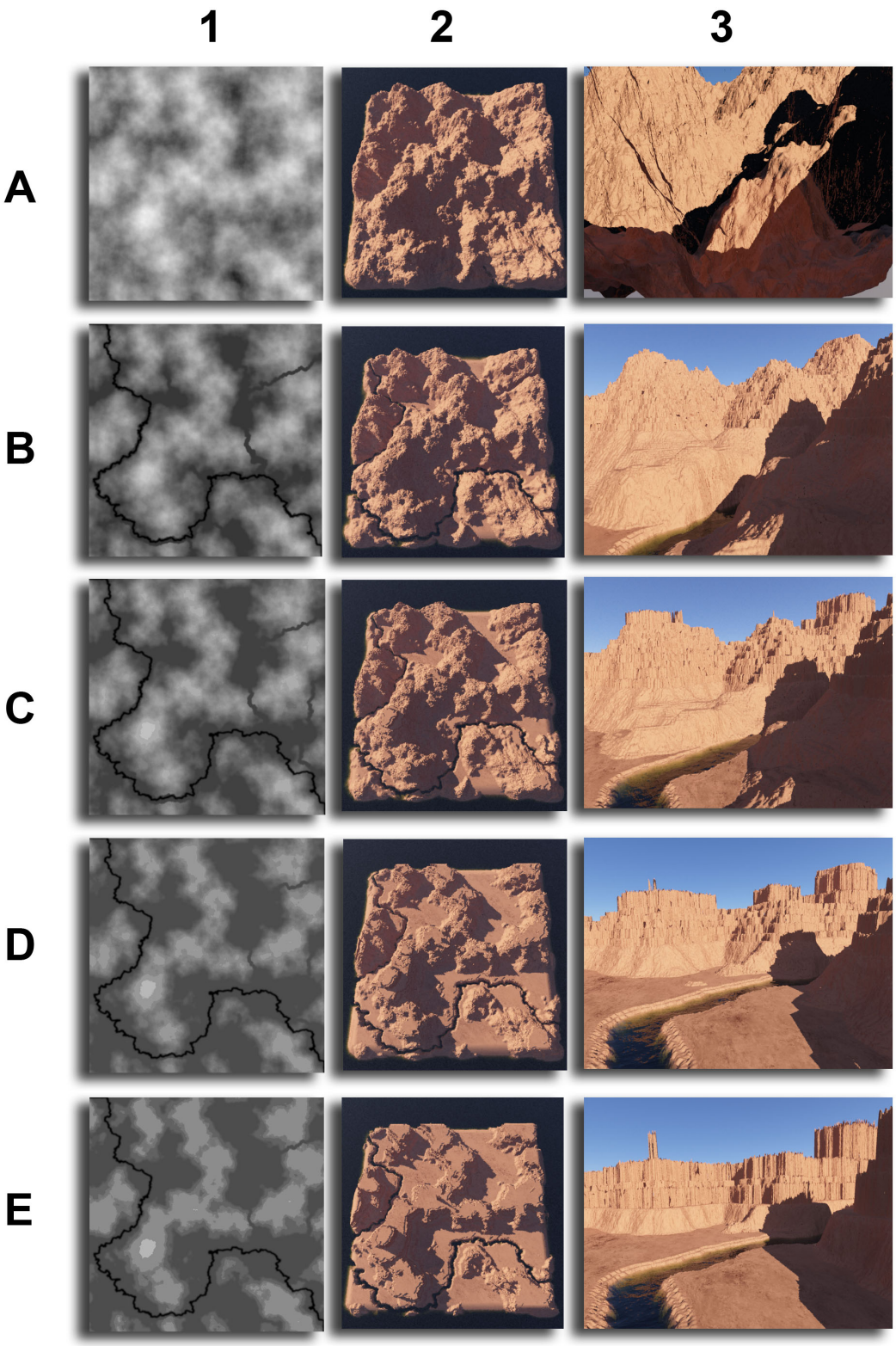

Figura 4.13: Influência da escolha do raio do *Mean Shift* no terreno gerado.

4.3.2 Realismo e Escala

Segundo GEORGE e HUGH (2006), o conteúdo gerado proceduralmente deve ter proporções relacionadas e equivalentes ao mundo real, além de poder ser ampliada ou diminuída. Essa característica é definida pela escala. Já o realismo está associado ao fato do conteúdo gerado se parecer com o mundo real. Logo, os resultados são apresentados de maneira visual e levam em consideração a avaliação subjetiva das imagens do mundo real. Os resultados foram divididos em dois grupos e apresentados nas Figuras 4.14 e 4.15.

As imagens apresentadas nesta seção mostram as características que foram apresentadas na Figura 1.3. Observam-se nas imagens geradas das Figuras 4.14 e 4.15 a existência dos planaltos, penhascos, artefatos, sopé da montanha (com declive suave), planícies e o curso do rio. Apesar de todos os cânions terem essas mesmas características em maior ou menor escala, a fim de facilitar a comparação, apresentam-se algumas imagem reais com os resultados obtidos em comparação com os gerados na lista a seguir.:

- **Planícies** — Cânions reais Figura 4.3 (*"Canyon de Chelly National Monument, Arizona, USA"* e *"Canyon de Chelly NM AZ"*) – Cânions gerados proceduralmente **Figuras 4.14 (A,B,C) e 4.15 (B)**;

- **Planaltos** — Cânions reais Figura 4.7 (*"Grand Canyon Helicopter Flight (20), Arizona, US"* – foto da esquerda) – Cânions gerados proceduralmente **Figuras 4.14 (C,D) e 4.15 (A,B,D)**;

- **Penhascos** — Cânions reais Figura 4.8 (*"Grand Canyon west"* e *"Santa Elena HDR, Brewster County, Texas, US"*) – Cânions gerados proceduralmente **Figuras 4.14 (E) e 4.15 (F)**;

- **Artefatos** — Cânions reais Figura 4.3 (*"Canyon de Chelly NM AZ"*– foto da direita), Figura 4.6 (*"Spider Rock, Apache County, Arizona, US"* – foto da esquerda) – Cânions gerados proceduralmente **Figura 4.14 (C,D,F)** ;

- **Sopé da montanha** — Cânions reais Figura 4.5 (*"Grand Canyon DEIS Aerial: Kwagunt Butte, Malgosa Crest, Nankoweap Mesa,Arizona, US"* e *"Canyon Overlook II, Utah, US"*), Figura 4.7 (*"First canyon, Cameron, Arizona, US"* – foto

da direita) – Cânions gerados proceduralmente **Figuras 4.14 (A,B,C,D,F) e 4.15 (B,C)**.

- **Curso do rio** — Cânion real Figura 4.10 (*Grand Canyon* visão aérea) – Cânions gerados proceduralmente **Figuras 4.14 (A,B) e 4.15 (A,B)**.

A tabela 4.2 apresenta as parametrizações utilizadas para a geração dos relevos apresentados nas Figuras 4.14 e 4.15. Para todos os testes utilizou-se um raio igual a 7, sendo que essa escolha está dentro da faixa que gera melhores resultados, conforme discutido na seção 4.3.1.

Tabela 4.2: Parâmetros de configuração das imagens das Figuras 4.14 e 4.15.

Figura	Letra	raio	*seed* (Perlin)	Áreas abertas	Área de declive
4.14	A	7	101	20%	40%
4.14	B	7	19	30%	50%
4.14	C	7	99	20%	40%
4.14	D	7	69	20%	40%
4.14	E	7	19	30%	10%
4.14	F	7	69	20%	40%
4.15	A	7	99	20%	40%
4.15	B	7	69	20%	40%
4.15	C	7	13	20%	40%
4.15	D	7	19	10%	50%
4.15	E	7	7	20%	40%
4.15	F	7	7	20%	40%

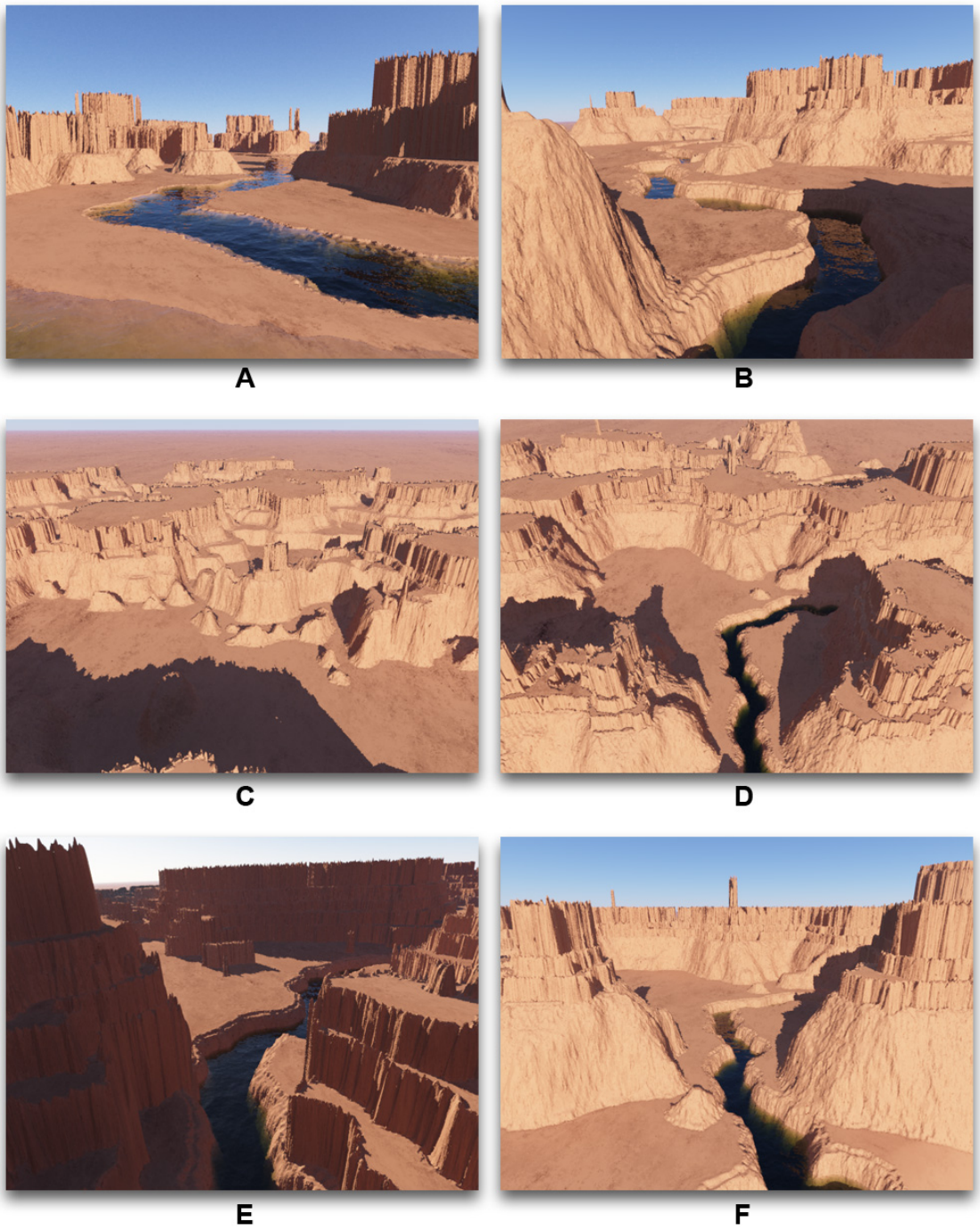

Figura 4.14: Primeiro grupo de imagens de cânions procedurais.

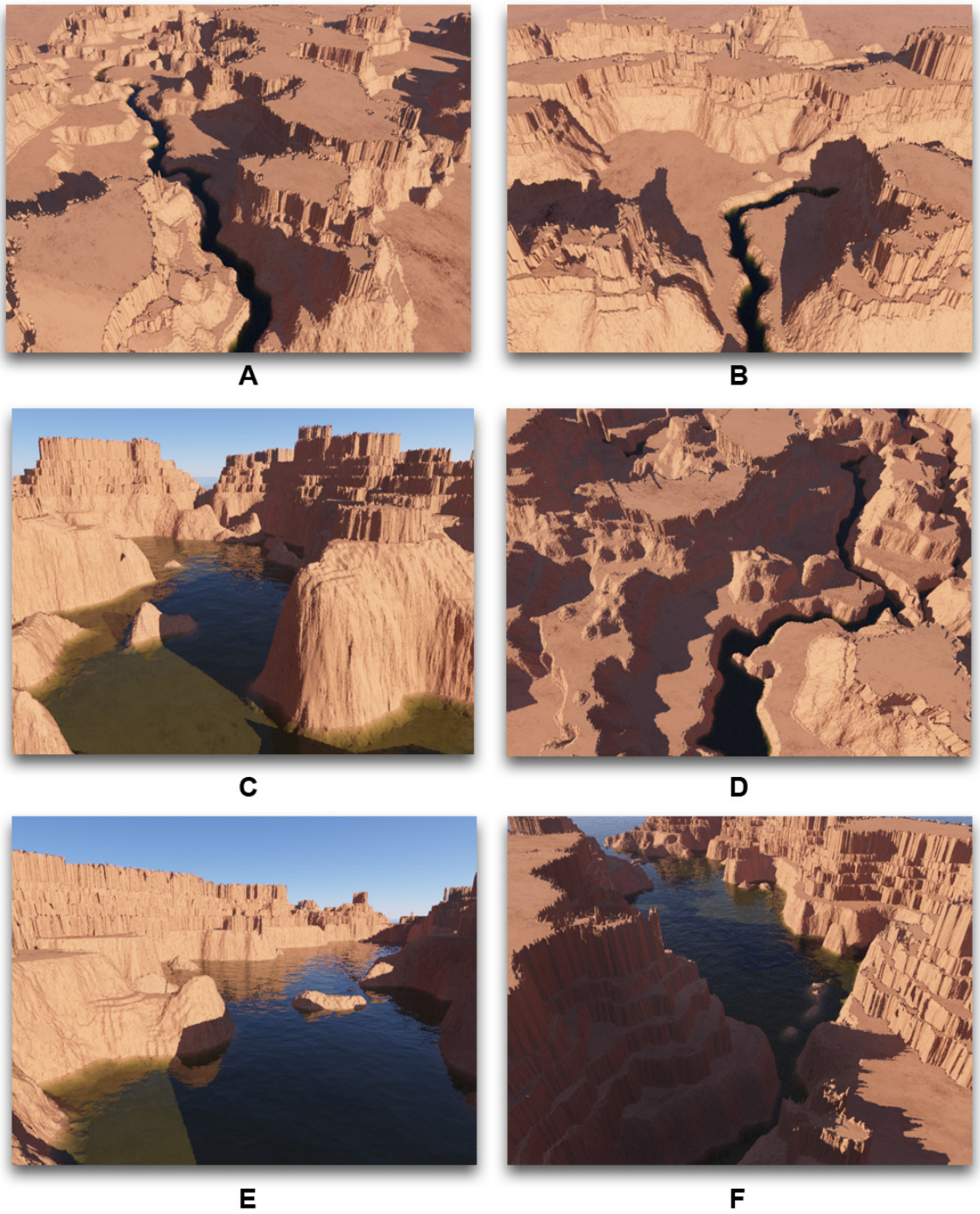

Figura 4.15: Segundo grupo de imagens de cânions procedurais.

4.3.3 Variações

Ainda segundo GEORGE e HUGH (2006), a variação é uma característica importante para os trabalhos relacionados a geração procedural. Acrescenta-se, ainda, o fato da natureza apresentar uma quantidade muito grande de formas e estruturas, sendo muitos os fatores que influenciam a formação geológica. As Figuras 4.14 e 4.15, já mencionadas, apresentam diversos resultados no contexto da visão de um personagem imerso no cenário.

As Figuras 4.16 e 4.17 – cujos os parâmetros são indicados na Tabela 4.3 – apresentam a visão aérea de diversos cenários gerados pela solução proposta. Nas imagens em questão, observa-se a existência de diversas planícies e a conexão delas entre si. Além disso, é possível perceber que os cenários apresentam grande variação entre si, o que torna os resultados bastante relevantes e interessantes no contexto de jogos digitais. Conforme mencionado anteriormente, o raio 7 está dentro da faixa dos melhores resultados e, por isso, ele foi escolhido para a geração dos cenários.

Tabela 4.3: Parâmetros de configuração das imagens das Figuras 4.16 e 4.17.

Figura	Letra	raio	*seed* (Perlin)	Áreas abertas	Área de declive
4.14	A	7	1	40%	40%
4.14	B	7	3	40%	40%
4.14	C	7	83	20%	40%
4.14	D	7	13	20%	40%
4.14	E	7	7	20%	40%
4.14	F	7	11	20%	40%
4.15	A	7	101	20%	40%
4.15	B	7	43	20%	40%
4.15	C	7	19	10%	10%
4.15	D	7	69	20%	40%
4.15	E	7	19	30%	10%

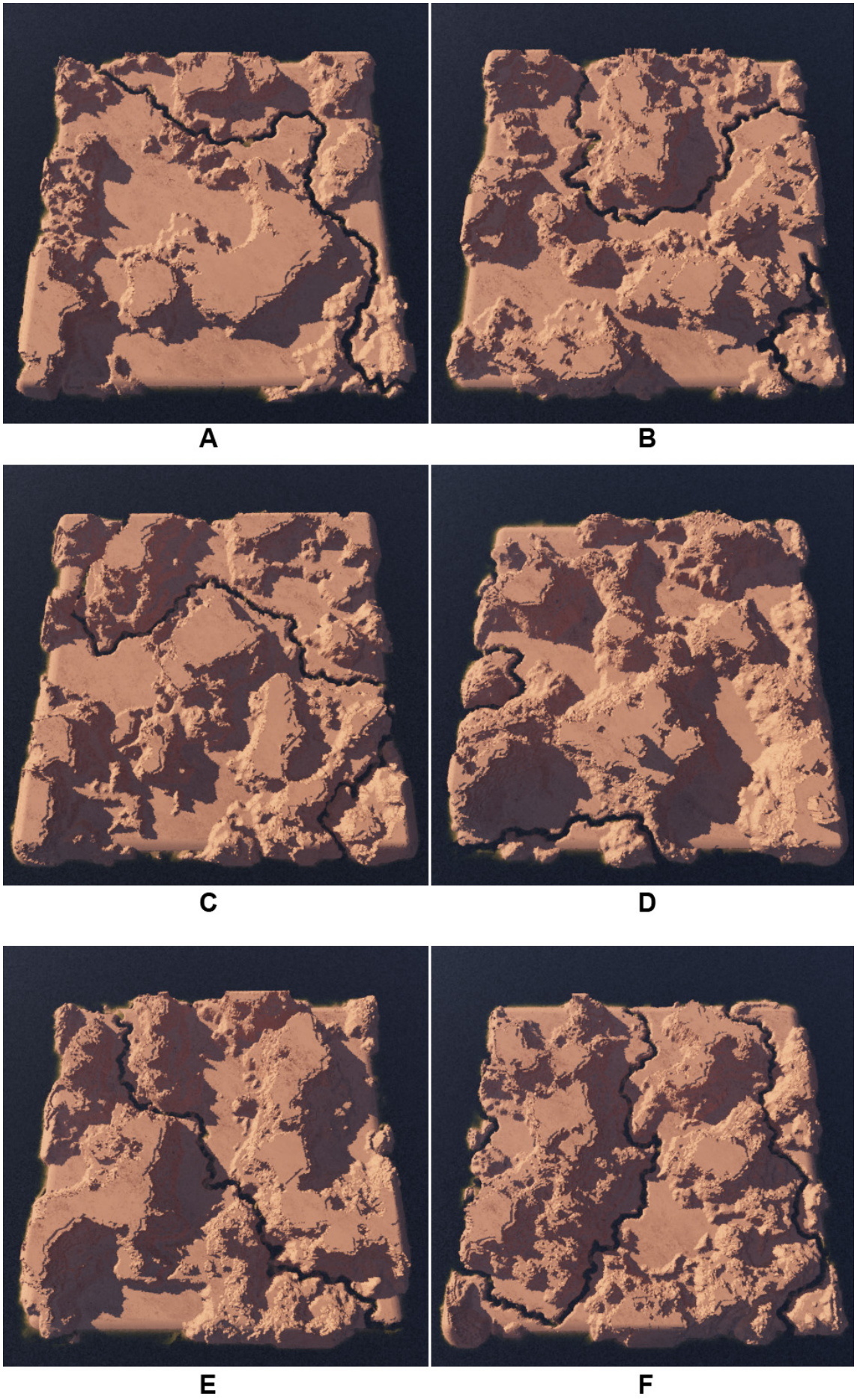

Figura 4.16: Primeiro grupo de imagens de cenários visto sob uma visão aérea.

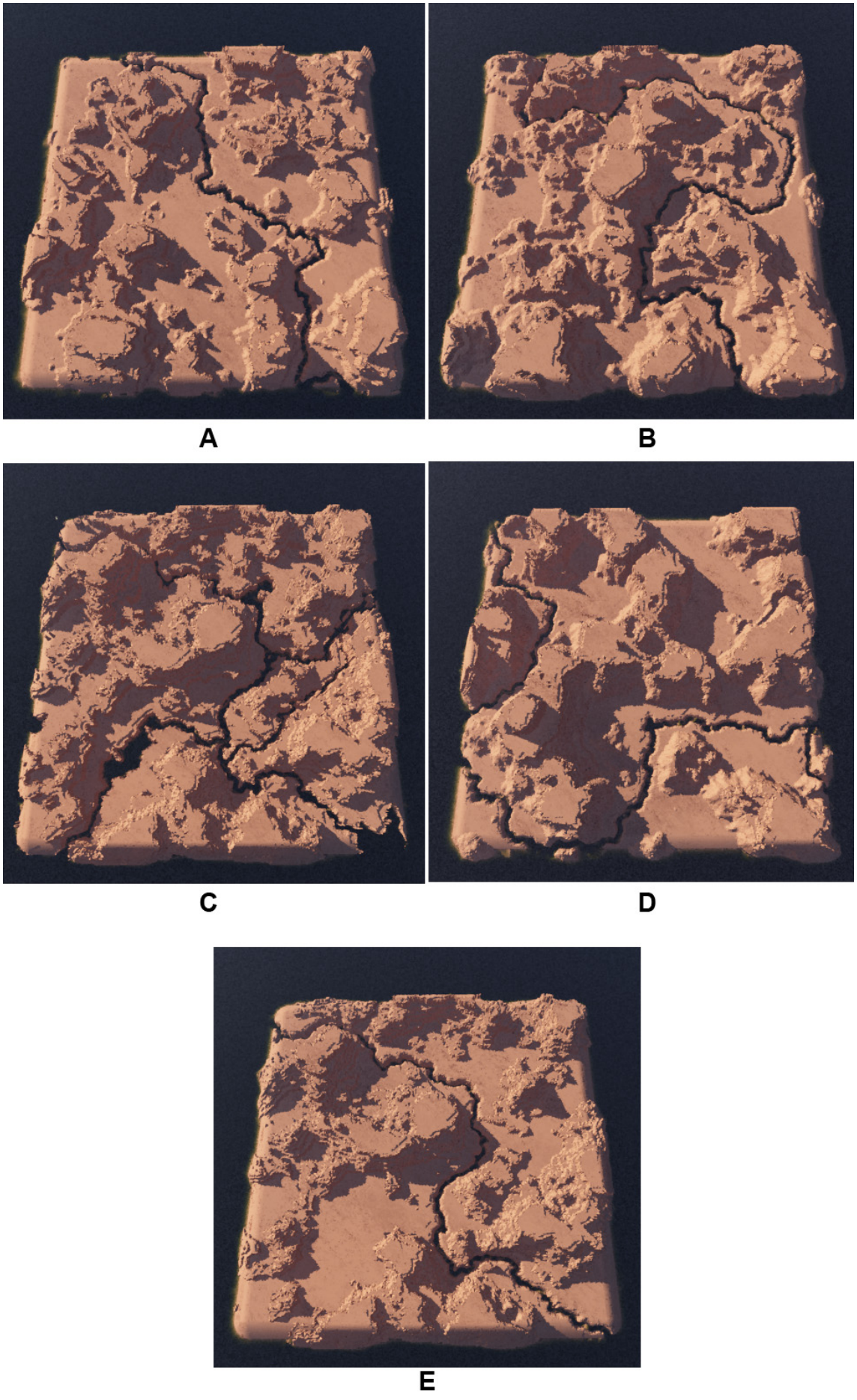

Figura 4.17: Segundo grupo de imagens de cenários visto sob uma visão aérea.

4.4 Teste de carga

O algoritmo foi implementado em C++ e utilizou a biblioteca *OpenCV* como base das operações de visão computacional e processamento de imagens. A implementação do método de clusterização *Mean Shift* foi realizada por SCHARDONG (2011). Já o algoritmo Dijkstra utilizado foi a partir da implementação de HENZ (2011).

Os testes foram realizados em um MacBook Pro com processador Core 2 Duo de 2457 MHz, placa gráfica Nvidia 320M (256 MB compartilhado da memória principal), 4 GB de memória ram DDR3 (1067 MHz) e sistema operacional Mac OS X 10.7.2 . A Tabela 4.4 apresenta a configuração de *hardware*. A Tabela 4.5, por sua vez, apresenta as principais informações sobre as bibliotecas utilizadas para a implementação e testes. As imagens utilizadas nos testes foram geradas na resolução de 1024x1024 e em 8 bits em tons de cinza.

Tabela 4.4: Tabela de configuração de *hardware* da máquina de testes

	Core 2 Duo	Nvidia 320 M
Núcleos	2	48
Clock por processador	2457 MHz	450 MHz
Threads	Em *software* (SO)	Em *hardware*

Tabela 4.5: Tabela de configuração de *software* da máquina de testes

	Versão	Compilação para 64 ou 32 Bits
OpenCL	1.50.63	32 e 64
OpenCV	2.2	32

Para avaliar o custo total das etapas dos algoritmos envolvidos nesta dissertação, o programa foi executado 22 vezes, variando o raio de entrada de 3 a 24 (o que fez o número de clusters identificados pelo *Mean Shift* variar de 56 a 1), considerando somente a abordagem sequencial em CPU. Testes que levam em consideração questões do impacto do uso da tecnologia OpenCL também são apresentados.

Para os testes que avaliam o custo total da aplicação, observou-se que as partes do programa que tinham maior consumo de recursos computacionais estavam associadas ao uso do *Mean Shift*, como no caso da definição das planícies (44.29%) e definição de regiões de penhascos (20,47%). Outro custo computacional bastante significativo observado foi no uso do algoritmo Dijkstra (sequencial) para execuções sucessivas, no caso da conexão

entre as planícies (30,32%). Essas informações podem ser observadas na Figura 4.18, que apresenta o gráfico dos principais custos computacionais associados ao método proposto.

O consumo de recursos computacionais associado ao algoritmo de rotulagem de componentes conectados de CHANG et al. (2004), através da biblioteca cvBlobs (CVBLOBS, 2012), foi bastante baixo, se comparado com o restante da aplicação, obtendo tempo médio de execução de 0,02 segundos, o que representa 0.04% do tempo total utilizado.

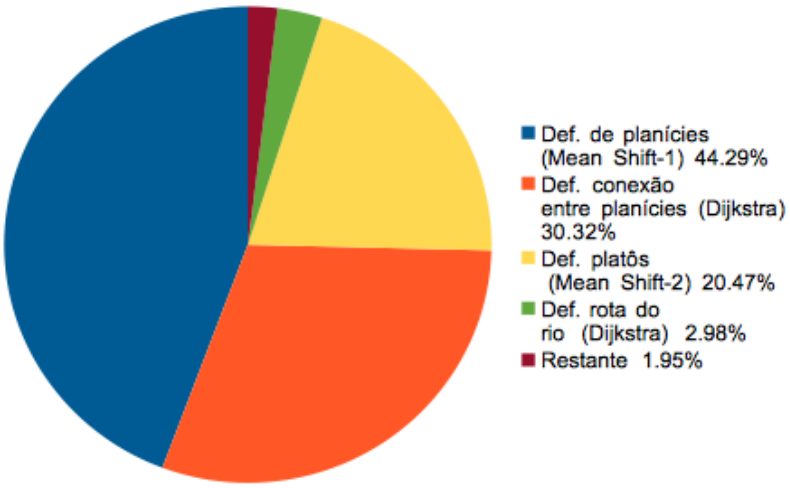

Figura 4.18: Porcentagem de tempo gasto por atividade.

A métrica de avaliação do impacto do uso da tecnologia OpenCL, no contexto desta dissertação, foi a execução sucessiva do algoritmo para diferentes valores do raio, que variou de 3 a 24, sendo 24 um raio tão grande que o algoritmo *Mean Shift* encontrou apenas um cluster (uma única planície). Assim sendo, executou-se o programa 66 vezes, sendo 22 vezes para OpenCL em GPU, 22 vezes para OpenCL em CPU, e as 22 vezes restantes em CPU sequencial.

O tempo médio total sequencial da aplicação para a geração de um mapa foi de 61 segundos, sendo 48,6 segundos o menor tempo, e 71,0 segundos o maior tempo obtido. Para o caso do programa rodando com OpenCL em GPU, o custo médio foi de 56,44 segundos, sendo 41,75 segundos o menor tempo, e 72,15 segundos o maior. No último caso, rodando com OpenCL em CPU, o custo médio foi de 71,25 segundos, sendo o menor tempo 55,51 segundos, e o maior de 82,9 segundos. O gráfico da Figura 4.19 apresenta os resultados obtidos pelo tempo de execução, sendo que, quanto menor o tempo obtido, melhor é o desempenho.

Dessa forma, observou-se que existe um pequeno ganho de desempenho com o uso

de OpenCL em GPU, se comparado com a execução sequencial (7,65%). Já no caso do uso do OpenCL em CPU, observou-se uma perda de desempenho médio de 16,66%, comparando-se com a execução sequencial. Acredita-se que essa perda de desempenho se deva ao *overhead* associado ao controle e restrições que o sistema tem que gerenciar para ter a estrutura de *Threads* funcionando em paralelo. Ou seja, a natureza da imagem segmentada não conseguiu tirar proveito da implementação do *Mean Shift* com o uso do OpenCL em CPU.

Outra questão observada foi o impacto do uso da tecnologia OpenCL em código sequencial após o código paralelizado ter sido executado. Dentre os testes, notou-se que o algoritmo de Dijkstra, que define o curso do rio, teve um acréscimo de 11,54% para o OpenCL em GPU, e de 11,33% para o OpenCL em CPU. Uma questão importante a destacar é que o algoritmo de Dijkstra foi implementado exclusivamente para a execução sequencial e para todos os testes ele, teoricamente, deveria apresentar uma variação bastante baixa ou insignificante. Apesar dessa constatação, não se tem informações e conhecimento das razões do OpenCL impactar em algoritmos sequencias que não usem essa tecnologia. Assim sendo, sugere-se como trabalho futuro um estudo mais aprofundado do impacto do uso da tecnologia OpenCL em algoritmos sequenciais de execução posterior ao código paralelizado.

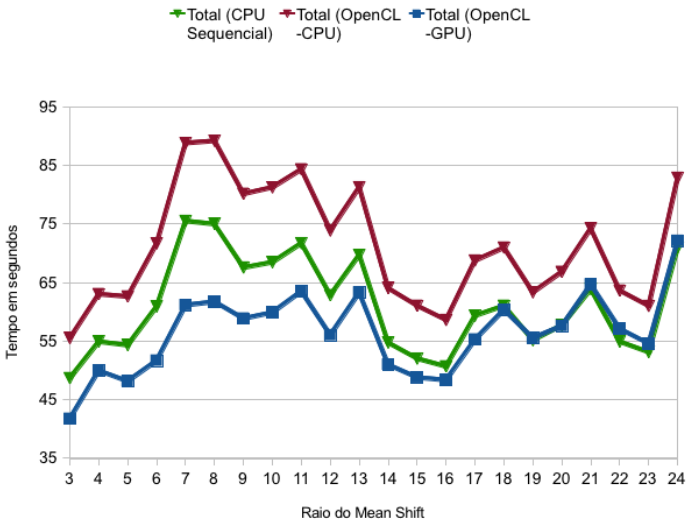

Figura 4.19: Gráfico comparativo de tempo de execução. Compara-se o impacto do uso do OpenCL no *Mean Shift* dentro do contexto do algoritmo proposto, tanto para GPU, quanto para CPU com relação ao algoritmo sequencial.

5 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresentou um método para a geração de cenários de cânions com foco em jogos digitais. O método se mostrou bastante eficiente e capaz de gerar o relevo com bastante variação. Apesar do método ser não assitido, o usuário tem controle sobre o relevo através de diversas parametrizações, o que permite a geração do terreno com maior ou menor quantidade de planícies, penhascos e caminhos. O usuário pode, ainda, definir a região de declive suave nos sopés das montanhas.

Para a definição do rio, o algoritmo de Dijkstra foi aplicado sobre um reticulado criado a partir do ruído de Perlin. A função de custo criada baseia-se no princípio da mínima ação e pode resultar em custos negativos, sendo que o algoritmo de Dijkstra não foi projetado para trabalhar sobre custos de arestas menores ou iguais a zero. Isso, a princípio, parece ser um impedimento, entretanto, apesar do custo de transição poder assumir valores menores que zero (o que indicaria uma queda), a abordagem proposta apresentou resultados satisfatórios para diversos testes, gerando curso complexos e interessantes e parecidos com o do rio Colorado.

Outra técnica que apresenta grande influência nos resultados gerados é o algoritmo *Mean Shift*. Nesse sentido, através da escolha do raio tem-se a quantidade de segmentos que o algoritmo irá encontrar, o que tem impacto direto sobre a formação do relevo, podendo variar de pequenos degraus até grandes penhascos. De maneira geral, a escolha do raio é subjetiva e depende do usuário. Contudo, foram feitos vários experimentos com o raio variando entre 4 e 9, que geraram resultados satisfatórios. Dessa forma, como visto no comparativo com as imagens reais, o algoritmo *Mean Shift*, juntamente com o ruído de Perlin, se mostrou uma técnica muito apropriada para a geração de cenários de cânions.

A solução proposta também se mostrou muito eficiente. Pode-se gerar um mapa de altura em pouco mais de 1 minuto de processamento. Quanto considerado o envio de cenários para jogos pela rede, esse fato é consideravelmente um grande fator de compressão. Ou seja, não é necessário o envio de todo o cenário, mas sim apenas os parâmetros para a construção do terreno. Nesse sentido, acaba não sendo necessário o envio do cenário completo para diversos jogadores, pois, com os parâmetros e o baixo tempo de execução, a aplicação pode gerar o cenário do cânion deseja. Os jogos que fizer uso dessa abordagem podem, também, armazenar o cenário processado e, sempre que receber os mesmos

parâmetros, simplesmente carrega-los.

Esta dissertação também apresentou uma análise sobre o uso da tecnologia OpenCL no algoritmo *Mean Shift*, no contexto da síntese procedural de cânions. Nesse caso, evidenciou-se um pequeno ganho de desempenho para o caso da execução em GPU (7,65%) e uma perda considerável de performance para a execução em CPU paralelizada com OpenCL (16,66%). Dessa forma, considera-se o desempenho da tecnologia OpenCL abaixo do esperado. Isso se deve pelo *overhead* associado à comunicação com a memória e estruturas de controle.

Como resultado secundário, notou-se, ainda, que o código sequencial do algoritmo de Dijkstra, executado posteriormente ao *Mean Shift* em GPU, apresentou uma perda média em torno de 11% de desempenho, o que no contexto desta dissertação não representa fração de tempo significativa. Contudo, sugere-se como trabalho futuro uma análise aprofundada do impacto do uso da tecnologia OpenCL em código sequencial de posterior execução. Assim sendo, deve-se verificar se essa questão levantada foi um caso isolado ou se as características de processamento e memória cache tem influência sobre o resultado observado.

O trabalho desenvolvido apresentou diversas contribuições. De início, a análise dos cânions reais permitiu a criação de dois diagramas, recursivo e não recursivo, que representam as características importantes a serem replicadas em métodos procedurais para cânions. Outra contribuição foi a introdução do algoritmo *Mean Shift* e o de rotulagem de componentes conectados na área de síntese procedural. Por fim, criou-se um método completamente automatizado capaz de gerar uma quantidade muito grande de relevos de cânions, sendo essa última o objetivo principal e a principal contribuição.

A fim de aumentar o realismo é necessário trabalhar com questões que ficaram fora do escopo desta dissertação. Essas questões são sugeridas como trabalhos futuros e são listadas na sequência:

- Geração de texturas realistas para serem aplicadas ao mapa de altura gerado, que possivelmente usem a técnica de *bump mapping*, e que levem em conta a natureza dos cânions, como, por exemplo, faixas horizontais de mesmas características;
- Geração de rotas alternativas para conectar os vértices do grafos dos centróides das planícies, gerando caminhos alternativos, melhorando ainda mais a aplicabilidade dos resultados em jogos digitais;

- Criação de uma função de custo parametrizável para, assim, poder ter maior controle sobre o formato das rotas encontradas, gerando maior variação;
- Desenvolver abordagens para a criação das planícies a partir do curso;
- Desenvolvimento de técnicas que possam fazer a distribuição de vegetação no cenário;
- Aprofundar o estudo nos processos erosivos a fim de gerar simulações para a criação de cenários de cânions;
- Desenvolvimento de técnicas para a criação de rochas soltas e sua posterior distribuição no terreno;
- Avaliar a aplicabilidade do método de segmentação *K-Means* para a geração procedural de terreno de cânions;
- Avaliar outras técnicas para a geração do reticulado inicial comparando-as com o resultado obtido com o ruído de Perlin.

REFERÊNCIAS

AL_HIKESAZ. **Wotan's Throne in the rain - Grand Canyon North Rim, Arizona, US.** (Foto) Disponível em: http://www.flickr.com/photos/alanenglish/3868924713/. Ano: 2009. Aceso em dezembro de 2011.

BALVARIUS. **Chiricahua Mountains, Cave Creek Canyon, Portal, Arizona, US**. (Foto) Disponível em: http://www.flickr.com/photos/balvarius/.3500607851/. Ano: 2009. Aceso em dezembro de 2011.

BENES, B.; FORSBACH, R. Layered Data Representation for Visual Simulation of Terrain Erosion. In: SPRING CONFERENCE ON COMPUTER GRAPHICS, 2001. **Anais...** [S.l.: s.n.], 2001.

BEVILACQUA, F. **Ferramenta para geração em tempo real de bordas de mapas virtuais pseudo-infinitos para jogos 3D**. 2009. Dissertação (Mestrado) — Universidade Federal de Santa Maria.

BEVILACQUA, F.; POZZER, C. T.; ORNELLAS, M. C. d. Charack: tool for real-time generation of pseudo-infinite virtual worlds for 3d games. **VIII Brazilian Symposium on Computer Games and Digital Entertainment**, [S.l.], 2009.

CHANG, F.; CHEN, C.-J.; LU, C.-J. A linear-time component-labeling algorithm using contour tracing technique. **Comput. Vis. Image Underst.**, New York, NY, USA, v.93, p.206–220, February 2004.

CHEN, G.; ESCH, G.; WONKA, P.; MüLLER, P.; ZHANG, E. Interactive procedural street modeling. **ACM Trans. Graph.**, New York, NY, USA, v.27, p.103:1–103:10, August 2008.

CHENG, Y. Mean Shift, Mode Seeking, and Clustering. **IEEE Trans. Pattern Anal. Mach. Intell.**, Washington, DC, USA, v.17, p.790–799, August 1995.

CHIBA, N.; MURAOKA, K.; FUJITA, K. An erosion model based on velocity fields for the visual simulation of mountain scenery. **Journal of Visualization and Computer Animation**, [S.l.], v.9, p.185–194, 1998.

COBALT123. **Coal Mine Canyon, Arizona**. (Foto) Disponível em: http://www.flickr.com/photos/cobalt/4580219355/in/photostream/. Ano: 2010. Acesso em agosto de 2011.

COMANICIU, D.; MEER, P. Mean Shift Analysis and Applications. **Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on**, [S.l.], 1999.

CORMEN, T.; LEISERSON, C.; RIVEST, R. **Introduction to Algorithms**. [S.l.]: McGraw-Hill, 1999.

CROSSLEY, R. **Gran Turismo 5 budget put at $60m**. Disponível em: http://www.develop-online.net/news/33204/Gran-Turismo-5-budget-put-at-60m. Ano: 2009. Acesso agosto de 2010.

CURTIS, E. S. **E. S. Curtis (1904)**: canyon de chelly – navajo. seven riders on horseback and dog trek against background of canyon cliffs. (Foto) Disponível em: http://en.wikipedia.org/wiki/File:Canyon_de_Chelly,_Navajo.jpg, Ano: 1904 . Acesso em agosto de 2011.

CVBLOBS. **Blob extraction library**. Disponível em: http://opencv. willowgarage.com/wiki/cvBlobsLib, Acesso em janeiro de 2012.

DACHSBACHER, C. **Interactive Terrain Rendering**: towards realism with procedural models and graphics hardware. 2006. Tese (Doutorado) — Universität Erlangen–Nürnberg.

DE CARLI, M. D.; D'ORNELLAS, C. M.; POZZER, T. C.; BEVILACQUA, F. A survey of procedural content generation techniques suitable to game development. **X Simpósio Brasileiro de Games e Entretenimento Digital**, [S.l.], 2011.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, [S.l.], v.1, p.269–271, 1959.

DOLLINS, S. C. **Modeling for the Plausible Emulation of Large Worlds**. 2002. Tese (Doutorado) — Brown University, United States of America.

DORAN, J.; PARBERRY, I. Controlled Procedural Terrain Generation Using Software Agents. **IEEE Transactions on Computational Intelligence and AI in Games**, [S.l.], v.2, n.2, p.111–119, 2010.

EBERT, D. S.; MUSGRAVE, F. K.; PEACHEY, D.; PERLIN, K.; WORLEY, S. **Texturing and Modeling**: a procedural approach. 2rd.ed. San Diego, CA, USA: AP Professional, 1998.

ELIAS, H. **Perlin Noise**. Disponível em: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm. Ano: 2011. Acesso em agosto de 2011.

FELDMAN, B. **Mean Shift Clustering**. Disponível em: http://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering/content/MeanShiftCluster.m . Ano: 2006. Acesso em agosto de 2011.

FUKUNAGA, K.; HOSTETLER, L. The estimation of the gradient of a density function, with applications in pattern recognition. **Information Theory, IEEE Transactions on**, [S.l.], v.21, n.1, p.32 – 40, jan 1975.

GALIN, E.; PEYTAVIE, A.; MARÈCHAL, N.; GUÈRIN, E. Procedural Generation of Roads. **EUROGRAPHICS: Computer Graphics Forum**, [S.l.], v.29, 2010.

GANG, L.; GUANGSHUN, S. Procedural Modeling of Urban Road Network. In: INFORMATION TECHNOLOGY AND APPLICATIONS (IFITA), 2010 INTERNATIONAL FORUM ON, 2010. **Anais...** [S.l.: s.n.], 2010. v.1, p.75 –79.

GEORGE, K.; HUGH, M. A Survey of Procedural Techniques for City Generation. **ITB**, [S.l.], n.December, 2006.

GOMES, J.; VELHO, L. **Computação Gráfica**. [S.l.]: Instituto de Matemática Pura e Aplicada - IMPA, 1998. v.1.

GOOGLE. **Google Earth**. Disponível em: http://www.google.com/intl/pt-PT/earth/index.html. Acesso em novembro de 2011.

GREUTER, S.; PARKER, J.; STEWART, N.; LEACH, G. Real-time procedural generation of 'pseudo infinite' cities. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES IN AUSTRALASIA AND SOUTH EAST ASIA, 1., 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p.87–ff. (GRAPHITE '03).

GUERRA, A. T.; GUERRA, A. J. T. **Novo Dicionario Geologico-Geomorfológico**. [S.l.]: BERTRAND BRASIL, 2008.

HÄGGSTRÖM, H. **Real-time generation and rendering of realistic landscapes**. 2006. Dissertação (Mestrado) — University of Helsinki, Finlândia.

HART, P.; NILSSON, N.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. **IEEE Transactions on Systems Science and Cybernetics**, [S.l.], v.4, n.2, p.100–107, Feb. 1968.

HENZ, B. **Geração de caminhos em cenários 3D a partir de mapas de custo utilizando Mean Shift**. Monografia (Graduação) - Bacharel em Ciência da Computação - Universidade Federal de Santa Maria. 2011.

HEY, J. **Spider Rock Canyon de Chelly Arizona**. (Foto) Disponível em: http://www.flickr.com/photos/palojono/4804211191/. 2009.

HEY, J. **Spider Rock, Apache County, Arizona, US**. (Foto) Disponível em: http://www.flickr.com/photos/verismovita/458422170/. 2009.

HNAIDI, H.; GUERIN, E.; AKKOUCHE, S.; PEYTAVIE, A.; GALIN, E. Feature based terrain generation using diffusion equation. **Computer Graphics Forum (Proceedings of Pacific Graphics)**, [S.l.], v.29, n.7, p.2179–2186, 2010.

HUIJSER, R.; DOBBE, J.; BRONSVOORT, W. F.; BIDARRA, R. Procedural Natural Systems for Game Level Design. **Games and Digital Entertainment, Brazilian Symposium**, Los Alamitos, CA, USA, n.November, 2010.

KANNAN, S. **Grand Canyon west**. (Foto) Disponível em: http://www.flickr.com/photos/sivaprakash/2392924119/. Ano: 2008, Aceso em agosto de 2011.

KELLEY, A. D.; MALIN, M. C.; NIELSON, G. M. Terrain simulation using a model of stream erosion. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS, 1988. **Anais...** [S.l.: s.n.], 1988. v.22, p.263–268.

LAGAE, A.; LEFEBVRE, S.; COOK, R.; DEROSE, T.; DRETTAKIS, G.; EBERT, D.; LEWIS, J.; PERLIN, K.; ZWICKER, M. A Survey of Procedural Noise Functions. **Computer Graphics Forum**, [S.l.], v.29, n.8, p.2579–2600, 2010.

LAU, K. **Canyon Overlook , Utah, US.** (Foto) Disponível em: http://www.flickr.com/photos/mischiru/1929779385/. Ano: 2007. Aceso em dezembro de 2011.

LAU, K. **Canyon Overlook II, Utah, US.** (Foto) Disponível em: http://www.flickr.com/photos/mischiru/1945322728/. Ano: 2007. Acesso em agosto de 2011.

LINDA, O. **Generation of planetary models by means of fractal algorithms**. [S.l.]: Czech Technical University, 2007.

LINTERMANN, B.; DEUSSEN, O. A modelling method and user interface for creating plants. **Computer Graphics Forum**, [S.l.], 1998.

MACDONALD, R. **Grand Canyon Helicopter Flight (20), Arizona, US.** (Foto) Disponível em: http://www.flickr.com/photos/ronmacphotos/4080682303/. Ano: 2009. Acesso em agosto de 2011.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. **Fifth Berkeley Symp. on Math. Statist. and Prob.**, [S.l.], v.1, p.281–297, 1967.

MANDELBROT, B. **Fractals**: form, chance, and dimension. [S.l.]: Freeman, 1977. 365p. v.1.

MARIANI, A. C. **Algoritmo de Dijkstra para cálculo do Caminho de Custo Mínimo**. Disponível em: http://www.inf.ufsc.br/grafos/temas/custo-minimo/dijkstra.html . Acesso em fevereiro de 2012.

MARTINS, D. S. **Classificação de fases em imagens hiperespectrais de raios X característicos pelo método de agrupamento por deslocamento para a média**. 2012. Dissertação (Mestrado) — Universidade Federal de Santa Maria.

MEHLFÜHRER, C. **Horseshoe Bend, Glen Canyon Dam, Arizona**. (Foto) Disponível em: http://en.wikipedia.org/wiki/File:Grand_Canyon_Horse_Shoe_Bend_MC.jpg. Ano: 2008. Acesso em agosto de 2011.

MOON, T. **Santa Elena HDR, Brewster County, Texas, US.** (Foto) Disponível em: http://www.flickr.com/photos/tychomoon/121565928/. Ano: 2006. Acesso em agosto de 2011.

MUSGRAVE, F. K.; KOLB, C. E.; MACE, R. S. The synthesis and rendering of eroded fractal terrains. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS, 1989. **Anais...** [S.l.: s.n.], 1989. v.23, p.41–50.

NAGASHIMA, K. Computer generation of eroded valley and mountain terrains. **The Visual Computer**, [S.l.], v.13, p.456–464, 1998.

NEGAUNEE, G. P. of. **Canyon de Chelly National Monument, Arizona, USA. View from White House Overlook to northeast**. (Foto) Disponível em: http://en.wikipedia.org/wiki/File:Canyon_de_Chell_7-27-09_(2).JPG. Ano: 2009. Aceso em agosto de 2011.

OLIVIER. **Grand Canyon, from South Rim near Visitor Center**. (Foto) Disponível em: http://en.wikipedia.org/wiki/File:Grand_Canyon_colors.jpg. Ano: 2008, Acesso em agosto de 2011.

OLSEN, J. Realtime Procedural Terrain Generation - Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games. **Department of Mathematics And Computer Science IMADA University of Southern Denmark**, [S.l.], 2004.

PAPHIO. **First canyon, Cameron, Arizona, US.** (Foto) Disponível em: http://www.flickr.com/photos/paphio/316137358/. Ano: 2006. Acesso em agosto de 2011.

PERLIN, K. An image synthesizer. **Annual Conference on Computer Graphics**, [S.l.], v.19, p.287–296, 1985.

PHELPS, J. M. **Glen Canyon**. (Foto) Disponível em: http://www.flickr.com/photos/mandj98/305947975/in/photostream/. Ano: 2004. Acesso em agosto de 2011.

PIMENTA, V. **Canyon Itaimbézinho**. (Foto) Disponível em: http://www.flickr.com/photos/valdiney/1392545393/. Ano: 2006. Acesso dezembro de 2011.

PLANETSIDE, S. **Terragen 2**. Disponível em: http://www.planetside.co.uk/content/view/15/27/. Acesso dezembro de 2011.

POV-RAY. **Height Field**. [S.l.: s.n.], 2011. Disponível em: http://www.povray.org/documentation/view/3.6.1/279/. Acesso em dezembro de 2011.

PRUSINKIEWICZ, P.; HAMMEL, M. A Fractal Model of Mountains with Rivers. In: PROCEEDING OF GRAPHICS INTERFACE '93, 1993. **Anais...** [S.l.: s.n.], 1993.

PRUSINKIEWICZ, P.; LINDENMAYER, A. **The algorithmic beauty of plants**. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

ROSA, M. **Segmentação de grãos de hematitaem amostras de minério de ferro por análise de imagens de luz polarizada**. 2008. Dissertação (Mestrado) — Mestrado em Engenharia de Produção - Universidade Federal de Santa Maria. 2008.

ROUDIER, P.; PEROCHE, B.; PERRIN, M. Landscapes Synthesis Achieved through Erosion and Deposition Process Simulation. **Computer Graphics Forum**, [S.l.], v.12, p.375–383, 1993.

SCHARDONG, G. G. **Acelerando o Mean Shift com OpenCL**. Monografia (Graduação) - Bacharel em Ciência da Computação - Universidade Federal de Santa Maria. 2011.

SERVICE, N. P. **Grand Canyon DEIS Aerial**: kwagunt butte, malgosa crest, nankoweap mesa,arizona, us. (Foto) Disponível em: http://www.flickr.com/photos/grand_canyon_nps/5476820919/. Ano: 2010. Acesso em agosto de 2011.

SHAPIRO, L.; STOCKMAN, G. **Computer Vision**. [S.l.]: NJ: Prentice-Hall, 2001.

SILVA, E. L. da; MENEZES, E. M. **Metodologia de Pesquisa e Elaboração de Dissetação**. Florianópolis: Laboratório de Ensino a Distância da UFSC, 2001.

SIMMER, I. D. **Bryce Canyon, Utah, US**. (Foto) Disponível em: http://www.flickr.com/photos/blogography/607548544/. Ano: 2004 .Acesso em agosto de 2011.

SIWEK, S. E. **Video games in the 21st century**: the 2010 report. [S.l.]: ESA: Entertainment Software Association, 2010.

SONG, G.-S.; HSU, S.-K. Automatic extraction of ridge and valley axes using the profile recognition and polygon-breaking algorithm. In: **Computers and Geosciences**. [S.l.]: Elsevier, 1998.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introdução ao Data Mining**: mineração de dados. [S.l.]: Ciência Moderna, 2009.

THIERRY. **Grand Canyon**. (Foto) Disponível em: http://www.flickr.com/photos/http2007/4705776152/. 2010.

WALSH, P.; GADE, P. Terrain generation using an Interactive Genetic Algorithm. In: EVOLUTIONARY COMPUTATION (CEC), 2010 IEEE CONGRESS ON, 2010. **Anais...** [S.l.: s.n.], 2010. p.1 –7.

WELLER, D.; WELLER, S. **Canyon de Chelly NM AZ**. (Foto) Disponível em: http://www.flickr.com/photos/21725375@N06/4507374444/. Ano: 2005, Acesso em agosto de 2011.

WIKIPEDIA. **Canyon**. Disponível em: http://en.wikipedia.org/wiki/Canyon. Acesso em dezembro de 2011.

ZHOU, H.; SUN, J.; TURK, G.; REHG, J. M. Terrain Synthesis from Digital Elevation Models. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.13, p.834–848, 2007.