INSTITUTO SUPERIOR
DE TECNOLOGIAS
AVANÇADAS

Técnicas Procedimentais e Computação Gráfica na Criação de Conteúdo Digital

Luís Pedro Sousa Morgado, Nº 21419

3º Ano – Laboral, Instituto Superior de Tecnologias Avançadas

Licenciatura em Engenharia Multimédia, Lisboa

Coordenador: Professor Dr. Pedro Brandão

Orientador: Professor Dr. Paulo Duarte

2020/2021

Agradecimentos

Agradeço à professora Dulce Mourato pelo apoio e disponibilidade durante as várias etapas da elaboração deste projeto.

Resumo

O objeto do presente estudo consiste em analisar as potencialidades da utilização de técnicas procedimentais e computação gráfica na criação de conteúdo digital.

Este trabalho explora a utilização de técnicas procedimentais, referindo o que são, como surgiram e como são utilizadas em simultâneo com a computação gráfica.

A pesquisa de diferentes formas e aplicações de técnicas procedimentais teve como foco a sua utilização em modelos tridimensionais (3D).

Por último, estas técnicas foram aplicadas através da criação de geradores procedimentais, automatizando o processo de desenvolvimento de conteúdo digital 3D.

Palavras-chaves: Geração Procedimental, 3D, Modelos Tridimensionais, Métodos Generativos, Automatização.

Abstract

The object of this study is to understand the potential of using procedural techniques and computer graphics in the creation of digital content.

This work explores the use of procedural techniques, referring to what they are, how they emerged and how they are used simultaneously with computer graphics.

The research on different forms and applications of procedural techniques focused on their use in three-dimensional (3D) models.

Finally, these techniques were applied through the creation of procedural generators, automating the 3D digital content development process.

Keywords: Procedural Generation, 3D, Three-dimensional Models, Generative Methods, Automation.

Índice

Índice de figuras

Introdução

A criação de conteúdo digital e a evolução tecnológica têm crescido a uma grande velocidade. Os consumidores são cada vez mais exigentes e a concorrência é cada vez maior.
A demanda pela criação de conteúdo digital teve um grande desenvolvimento nos últimos 20 anos. Fatores como o fácil acesso a computadores e à Internet contribuíram fortemente para o desenvolvimento desta área.

Com a criação de conteúdo digital de maior qualidade e em maior quantidade seria expectável que, com o decorrer do tempo, o custo de produção também crescesse de forma linear, no entanto, não é isso que se verifica. A utilização de técnicas procedimentais e de geradores procedimentais, enquanto forma de gerar conteúdo digital de modo automatizado, é um dos fatores que permite reduzir os custos de produção sem impacto significativo na qualidade do conteúdo criado.

Neste projeto explorei a utilização de técnicas procedimentais de forma a automatizar a criação de conteúdo digital, conciliando técnicas de programação e computação gráfica.
Procedi à exploração e aplicação destas técnicas, criando uma ferramenta que automatiza a criação de modelos tridimensionais (3D).

A escolha deste tema reflete o meu crescente interesse em aprofundar o conhecimento na área de computação gráfica. Apesar de algum conhecimento já adquirido, este projeto ajudou a compreender melhor o funcionamento das técnicas procedimentais, permitindo conhecer quão vasta é esta área.

O projeto é constituído por duas partes: a investigação, onde se encontra o estado da arte, e aplicação, onde se encontram os três exemplos práticos. O estado da arte apresenta, de uma forma simplificada e resumida, as bases para compreender as técnicas e o funcionamento dos geradores procedimentais. Os exemplos práticos demonstram diferentes formas de utilizar os geradores procedimentais para criar modelos 3D.

Estado da Arte

Com este projeto pretendeu-se criar uma ferramenta que automatiza a criação de modelos tridimensionais (3D) e demonstra as potencialidades de utilização de técnicas procedimentais no âmbito da computação gráfica, para quem pretenda desenvolver e conhecer como automatizar o processo de desenvolvimento de conteúdo digital.

Neste capítulo será realizada uma revisão bibliográfica com o objetivo de perceber as potencialidades das técnicas procedimentais da criação de conteúdo digital, tendo como foco a criação modelos 3D e respondendo às seguintes questões de investigação:

- O que são técnicas procedimentais?
- Em que áreas são utilizadas?
- Que ferramentas são utilizadas?
- Como funcionam?
- Como implementar?
- Quais as potencialidades na sua utilização e porque se usam?
- Quais os problemas e riscos na sua utilização?

História das técnicas procedimentais

Antes de mais, é necessário perceber como surgiram as técnicas procedimentais e como evoluíram no âmbito da ciência da computação.

Técnicas procedimentais segundo Ebert (2003, p.1) "Procedural techniques are code segments or algorithms that specify some characteristic of a computer-generated model or effect".

As técnicas procedimentais surgiram inicialmente na área da programação nos anos 60 como programação procedimental. Este tipo de programação consiste na execução de um conjunto de passos computacionais de forma sequencial para atingir o resultado pretendido. Os principais pontos chaves da programação procedimental são a sua composição sequencial, a sua modularidade e a sua capacidade de repetição. Faz parte de um dos três paradigmas da programação, a par da programação orientada a objetos e a programação funcional.

Mais tarde, as técnicas procedimentais surgiram como uma forma de salvar espaço no disco, num tempo onde era necessário salvar memória para caber em disquetes e cartões de memória. Desenvolvedores de videojogos criaram, assim, uma maneira de fornecer muito conteúdo ao utilizador final, gerando conteúdo de forma procedimental, à medida que o utilizador ia jogando. Os primeiros exemplos desta implementação verificaram-se na geração de *dungeons*, monstros e itens em *Rogue* (1983) e um espaço infinito em Elite (1984).

No entanto, foi na área da computação gráfica que teve maior impacto histórico. Em meados da década 80 do século passado, as técnicas procedimentais foram utilizadas primeiramente para criar texturas para objetos, com a introdução de texturas 3D por Ken Perlin, Darwyn Peachey e Geoffrey Gardner em 1985 (Ken Perlin ganhou um Prémio de Realização Técnica da Academia de Artes e Ciência Cinematográfica em 1997 pelo desenvolvimento da sua função de ruído "Perlin Noise").

As técnicas procedimentais ganharam uma vasta gama de utilizações na criação de texturas realistas como madeira, mármore, rochas e nuvens.

O rápido desenvolvimento de técnicas procedimentais expandiu-se para a modelação 3D, criando assim os primeiros modelos computacionais de água, fumo, vapor, fogo e planetas. Estas técnicas também permitiram a animação e a simulação de fenómenos naturais, demonstrando-se como uma ferramenta poderosa na criação de movimentos complexos e realistas.

O desenvolvimento da linguagem de sombreamento (*shader language*) RenderMan, criada pela Pixar em 1989, expandiu a utilização de técnicas procedimentais, sendo ainda hoje uma das ferramentas de renderização 3D mais utilizadas pela indústria.

O surgimento de unidades de processadores gráficas (GPU) programáveis de baixo custo, o fácil acesso e o rápido aumento do poder computacional tornaram as técnicas procedimentais vitais para a criação de efeitos de alta qualidade em jogos de computador e entretenimento interativo (aplicações em tempo real).

Atualmente, a modelação, a texturização, a animação e o sombreamento procedimentais são ferramentas essenciais do processo de criação de conteúdo digital. São utilizadas nas aplicações gráficas realistas e animações, desde os efeitos visuais (VFX) nos filmes até aos jogos de computador.

Definições

Conteúdo digital pode ser definido por todo e qualquer conteúdo que existe no formato digital. Normalmente refere-se a filmes, jogos, músicas e imagens que estão disponíveis para distribuição em média eletrónica.

No processo de criação destes conteúdos de forma procedimental é necessário o desenvolvimento de geradores.

Entende-se por Gerador Procedimental, ou Procedural Generation (PG), o processo de criação de conteúdo através de algoritmos lógicos que utilizam técnicas procedimentais a partir de dados.

Outro conceito também muito referido na área da computação gráfica é o Gerador de Conteúdo Procedimental, ou Procedural Content Generation (PCG). Sendo um subgrupo da PG, pode definir-se como “the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces” segundo PCG Wiki. Desta forma, é possível criar planetas, dungeons, músicas, histórias, mundos e animações e incorporar num ambiente de jogo onde cada utilizador vai ter uma interação e experiência diferente.

A maior diferença entre PG e PCG não são tanto os algoritmos utilizados, mas mais como o conteúdo é usado depois de gerado. Os PCG são necessariamente dinâmicos e específicos de ambiente de jogos enquanto os PG são mais genéricos e permitem uma versatilidade de utilizações, englobando áreas como a arte, música, design e outros ramos generativos, não se restringindo apenas a jogos.

Contudo, a definição destes termos é um pouco difusa, não sendo possível apontar com rigor, os seus limites. Por exemplo, alguns autores preferem usar o termo “generative methods” para se referir a PG (Compton et al., 2013).

Embora muitas técnicas procedimentais sejam comuns tanto para a PG quanto para PCG, vou dar maior ênfase à PG ao longo deste projeto.

Áreas de Aplicação de Geradores Procedimentais

A área de Geração Procedimental é muito vasta e diversificada, sendo utilizada em diferentes contextos, de diferentes modos e com diferentes objetivos. Assim, enumero alguns exemplos e os seus principais focos.

Procedural Textures and Shaders

Procedural Textures tem como foco a criação de texturas usando um algoritmo em vez de dados armazenados diretamente. Este tipo de texturas, que pode ser 2D ou 3D, é frequentemente utilizado para modelar superfícies ou representações volumétricas de elementos naturais/orgânicos como madeira, mármore, granito, metal e pedra. As texturas trabalham em conjunto com *Shader* de forma a produzir a renderização final. Exemplo como Shadertoy (Quilez & Jeremias) que permite criar e compartilhar shader por meio de WebGL.
"A shader is a bit of computer code that is commonly used to describe how a surface will be rendered" (Polycount, para. 1).

Procedural Modeling

Procedural Modeling tem como foco a criação de geometria 3D a partir de algoritmos baseados em um conjunto de regras, em vez de ser o utilizador a editar o modelo manualmente.
É frequentemente aplicado quando é inviável a criação de modelos 3D à mão, usando modeladores genéricos 3D. É utilizado para criação de estruturas orgânicas como a geração de plantas e terrenos (exemplos como Carpentier , Parberry), ou para a criação de estruturas rígidas como na arquitetura (exemplo como KellerDev).

Procedural Animation

Procedural Animation tem como foco a criação de movimento a partir de algoritmos, permitindo uma maior diversidade de ações do que poderia ser criado utilizando animações predefinidas (exemplo com de Solberg).

Este movimento, 2D ou 3D, é frequentemente utilizado para simular sistemas de partículas (fluidos, gases, fogo), para simulação de têxteis, movimentação de personagens (exemplo como de Rosen e Hecker), destruição de edifícios (exemplo como de Richter), entre outras.
"The general animation approaches presented can be used with any procedural model or texture" (Ebert, 2003, p.261).

Existem muitas outras áreas que aplicam a geração procedimental para criar conteúdo digital. Alguns exemplos como a geração de mapas (Wolverson), de histórias (Martens & Cardona), de diálogos (Paradis), de textos (Podolny) ou de músicas (Blob Opera).

Tipos de Geradores Procedimentais

Os geradores procedimentais podem ser separados em dois grandes grupos segundo Grendel Games (2018) :

- Offline/Development-time
- Online

Os geradores procedimentais Offline/Development-time criam conteúdo *offline*, como uma ferramenta de *design*. O conteúdo é guardado e carregado. Isto permite gerar conteúdo com mais detalhe, dado que a ferramenta de geração dispõe de mais tempo para processamento. Exemplos de software com Houdini (SideFX), Substance (Adobe), World Machine e SpeedTree permitem a criação de conteúdo procedimental altamente detalhado que depois pode ser exportado para um videojogo.

Os geradores procedimentais Online podem ser integrados dentro de um jogo ou aplicação interativa e usados para criar conteúdo em tempo de execução/tempo real. Isto permite uma maior interatividade com o utilizador final e uma experiência única para cada utilizador. Os geradores procedimentais Online segundo as apresentações “Introduction to Proceduralism” (Kruel, 2017) e “Working With Change” (McKendrick, 2018) podem ainda ser divididos em *Load Time*/*Ahead of Time* e *Run-time*.

A geração em *Load Time* consiste na criação de conteúdo antes de ser utilizado. O processo de geração ocorre no início ou durante uma tela de carregamento. Por exemplo, o videojogo Minecraft (Mojang Studios, 2011) onde um novo mundo é gerado antes do jogador começar a jogar.

A geração em *Run-time* consiste na criação de conteúdo em tempo real, à medida que o utilizador interage com o gerador. Todos estes processos lógicos têm de decorrer em poucos milissegundos. Por exemplo, se um jogo estiver a correr a 60 FPS, há apenas cerca de 16 milissegundos para executar todo o processo lógico. Exemplos de videojogos como No Man’s Sky (Hello Games, 2016) e Horizon Zero Dawn (Guerrilla Games, 2017) onde são abordados nas apresentações, respetivamente, “Continuous World Generation in ‘No Man's Sky’” (GDC, 2017) e “GPU-Based Run-Time Procedural Placement in ‘Horizon: Zero Dawn’” (GDC, 2017) e ferramentas como Tiltbrush (Google, 2016), conseguem processar

toda a lógica de criação de terreno, inteligência artificial, física, vegetação, renderização, entre outros, em poucos milissegundos por *frame*.

Ferramentas Procedimentais

Atualmente existe uma grande diversidade de ferramentas para a criação de conteúdo digital que utiliza geradores procedimentais. Em seguida, apresento algumas ferramentas que utilizam geradores procedimentais.

Houdini

Houdini é um software 3D criado pela SideFX, especialmente conhecido pelas suas capacidades procedimentais e estrutura Node-based. Este software pode ser utilizado para a modelação, animação, efeitos especiais, renderização, desenvolvimento de jogos entre outros. "Houdini is a very powerful 3D software application that has the ability to create procedural assets for Movies and games. It is more widely known in the movie industry but it is becoming more and more integrated into the game development pipeline" (SideFX, para. 1).

A apresentação da Entagma na FMX (2017) expõe bem as características na utilização do Houdini.

Em conjunto com o Houdini também existe o Houdini Engine, que funciona como um *plug-in* para integrar com outro aplicativos, disponibilizando uma solução não grafica para processamento e distribuição de tarefas.

Blender

Blender é um software 3D, livre e *open source.* "It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation" (Blender, para. 8).

Atualmente apenas apresenta um sistema procedimental Node-based para a criação de materiais, composição e texturas. O recente projeto "Everything Nodes" tem como objetivo expandir e unificar a utilização de Nodes para as restantes áreas como: a modelação procedimental (Geometry Nodes), sistemas de partículas, simulação e animação.

Substance Designer and Substance Painter

Substance Designer é um software de texturização e modelação 3D. “It generates textures from procedural patterns or by manipulating bitmaps inside a node graph as well as generates 3D models” (Substance 3D Designer Documentation, para. 1). Apresenta uma estrutura Node-based e é mais utilizado para sistemas procedimentais.

Substance Painter é um software de texturização e pintura de modelos 3D. Apresenta uma estrutura Layer-based e é mais utilizado para sistemas semi-procedimentais.

Nuke

Nuke é um software de composição digital 2D e 3D com uma estrutura procedimental Node-based, frequentemente utilizado para pós-produção de filmes e televisão.

Fusion

Fusion é uma ferramenta procedimental Node-based de composição digital que faz parte do software de edição de vídeo DaVinci Resolve.

Outras ferramentas que utilizam sistemas procedimentais:

- Speedtree: software de geração de vegetação e animação procedimental;
- Sceelix: software open source de modelação 3D procedimental;
- World Machine: software de geração de terreno;
- World Creator: software de geração de terreno em tempo real.

Programação

Ao nível da programação deixo aqui uma lista de repositórios de geração procedimental em diversas linguagens: Java, JavaScript, Python, C#, C++, Rust, entre outras.

GitHub: https://github.com/topics/procedural-generation

Interagir com Ferramentas

Quantos às ferramentas a utilizar para os geradores procedimentais, como no anterior tópico foi referido, podem ter diferentes formas de interagir com o utilizador.

Visual Node Based

Node-based recorre à utilização de nodes que executam operações como aritmética, geometria, processamento de imagem e composição de materiais em valores gráficos. Nodes fornecem um fluxo de trabalho procedimental não destrutivo, permitindo uma grande flexibilidade e uma utilização intuitiva.

Nesta área destaca-se o evento digital Nodevember (Dichelle & Rood), que decorre anualmente durante o mês de novembro e promove a utilização deste tipo de ferramentas para criar geradores procedimentais.

Visual Layer Based

Layer-based recorre à utilização de camadas como forma de organização e separação de funções.

Atualmente as ferramentas, como Substance Painter ou Adobe Premiere, apresentam uma estrutura Layer-based. São mais indicadas para geradores semi-procedimentais.

Programação

Ao nível da programação, independentemente da linguagem escolhida para criar o gerador procedimental, tem de se ter em conta o formato em que os dados são guardados. Tendo em vista facilitar a interpretação, acelerar o processo de *debugging* e a reutilização dos dados no gerador, são frequentemente utilizados formatos “human-readable” como JSON ou XML.

Métodos de Geração Procedimental

Agora que abordamos as vastas implementações de geradores procedimentais, importa saber como funcionam, como utilizar e que técnicas implementar. Desta forma, vou utilizar como base o modelo criado pelo artigo (Compton et al., 2013) para métodos generativos, que apresenta, de uma forma simples, as diferentes fases de geração de conteúdo ou artefacto, como é referido no artigo, e integrar com métodos aditivos e métodos subtrativos referidos por Kate Compton (2016) na criação de geradores procedimentais.

Numa primeira fase, um gerador necessita, geralmente, de um ou mais valores de entrada (*input*). Este pode consistir em "user-provided tuning values, static assets (such as 3D models, story segments, or audio samples), or higher-level inputs like partially configured artifacts or scripts" (Compton et al., 2013).
De seguida, estes inputs vão para o gerador, parte central de todo o processo generativo, onde são manipulados, compostos ou combinados para criar um conjunto de artefactos. Nesta fase, são utilizados métodos aditivos de forma a criar uma grande quantidade de dados.

Muitas vezes, um método generativo inclui um terceiro elemento, o crítico, que valida o artefacto proposto pelo gerador, podendo ser humano ou computacional. São utilizados, nesta fase, métodos subtrativos de forma a reduzir o espaço de resultados e conseguir encontrar o artefacto pretendido. "The critic can also give the generator an explicit accept/reject decision, a reward value, or a corrected artifact to accommodate generators which incorporate feedback" (Compton et al., 2013).

Os inputs, o gerador e o crítico podem ser compostos por sistemas menores, incluindo outros métodos generativos.

Métodos Aditivos

Quanto aos métodos aditivos podemos referir, de forma geral, os principais:

Tile

Os métodos baseados em tiles permitem a agregação de módulos diferentes, de tamanho idêntico, de maneira a criar um sistema complexo. Os artefactos criados são apenas conjuntos selecionados ou ordenados a partir de soluções pré-criadas de maneira diferente. Um dos exemplos mais simples e antigos desta utilização são os baralhos de cartas, onde as cartas são as mesmas, mas são ordenadas e selecionadas de forma diferente, permitindo um vasto espaço de possibilidades de jogo. Videojogos como *Civilization V* (2010) utilizam *tiles* de forma a proporcionar uma vasta gama de *gameplay* diferentes, dependendo da posição e ordem que o jogador escolhe.

"Tile-based methods work for problems that can be broken-up into small chunks where internal structure matters, but that can still create interesting (but not constraint-breaking) behavior when combined in different orders" (Compton, 2016, para. 22).

Grammars

Métodos baseados em grammars utilizam artefactos e repetição, de forma a criar novos artefactos. Grammars utilizam uma lista de regras que indica como construir uma estrutura hierárquica a partir de elementos mais simples. Um dos tipos de grammars mais comuns são os *L-Systems*, que permitem a criação de uma estrutura complexa, como árvores e edifícios, a partir de uma lista de instruções.

A utilização de grammars permite uma grande flexibilidade e diversidade de resultados. Um bom exemplo é o artigo RoboGrammar (Massachusetts Institute of Technology, 2020) que utiliza grammars para gerar uma grande diversidade de estruturas de robôs com diferentes tipos de conexões e tamanho de ligações.

Distribution

Métodos baseados em *distribution* é um dos métodos com uma estrutura mais simples. Consiste na disposição de elementos num espaço ou tempo, conciliando com a probabilidade de distribuição. Normalmente, não basta a utilização de distribuição *random* ou *pseudo-random*, dado que esta não é comum na natureza. Na natureza, a

distribuição é hierárquica, agrupada e espaçada, sendo mais comum a utilização de algoritmos de geração de ruído como *Perlin* e *Voronoi.*

Métodos baseados em distribuição podem ser utilizados em videojogos como *No Man 's Sky* e *Spore* como demonstram respetivamente, a apresentação “Building Worlds Using Math(s)” (Murray, 2017) e o artigo “Fast Object Distribution” (Willmott, 2007), para a distribuição de edifícios, vegetação e criaturas ou também para a geração de *dungeons* “Procedural Dungeon Generation Algorithm” (Adonaac, 2015).

Parametric

Métodos baseados em *parametric* utilizam parâmetros de forma a regular, ajustar ou definir um comportamento específico de um gerador procedimental. Isto permite uma variabilidade unidimensional fixa, onde não existe uma variação da estrutura do gerador.

Os métodos *parametric* estão limitados à estrutura do artefacto que regulam, como é referido por Kate Compton “You can see something ‘new’, but never something surprising” (Compton, 2016).

No entanto, são métodos particularmente utilizados em geradores interativos, onde o desenvolvedor pode definir quantos parâmetros úteis forem necessários para que o utilizador opere com eficácia. Como é referido por Eber, “This parametric control unburdens the user from the low-level control and specification of detail” (Ebert, 2003, p.2).

Exemplos de utilização de métodos *parametric*: um parâmetro que torna as montanhas mais rugosas ou mais suaves, os modificadores presentes no *Blender* permitem ajustar e combinar uma vasta gama de geradores já predefinidos, o ajuste de parâmetros num HDA (Houdini Digital Assets) ou *Morph Handles* que permitem criar uma vasta variedade de criaturas em jogos como *Spore* e *No Man 's Sky*.

Interpretive

Métodos baseados em *interpretive* são um dos mais utilizados, e consistem na execução de um algoritmo que processa dados em outros dados mais complexos. A partir de uma estrutura de dados simples construímos uma estrutura mais complexa.

Exemplos de alguns métodos *interpretive*: algoritmos de captura de movimento (*mocap*), algoritmos de criação de ruído como *Perlin*, algoritmos de triangulação de geometria com *Voronoi* ou *Delaunay*, *Metaballs*, *Fractals* e *Constructive Solid Geometry*.

Simulation/Rule Systems

Métodos baseados em *simulation* permitem a criação de um grande número de output utilizando regras simples. Muitos destes algoritmos são baseados em fenómenos que ocorrem na natureza.

Alguns dos algoritmos mais conhecidos baseados em *simulation* são *Ant Colony Optimization*, *Firefly Algorithm*, *Ragdoll Physics*, *Cellular Automata (*exemplo *Game of Life), Fluid, Particles e Crowds Simulation*.

Métodos Subtrativos/Modelos de Treino

Quanto aos métodos subtrativos podemos referir, de forma geral, os principais:

Seeds

Seeds consiste num número que vai ser utilizado para iniciar um gerador. Ao usar o mesmo valor para iniciar o mesmo gerador o resultado final será sempre o mesmo.

“No matter how complex the algorithm is, if you start at the same point, and perform the same operations, you're going to get the same results” (Green, 2016).

A utilização de seeds é uma forma de obter controlo sobre o espaço de resultados gerados. Jogos como Minecraft fazem a utilização de seeds, que podem ser definidas pelo utilizador ou criadas de forma aleatória, na criação de um mundo.

Generate and Test

Generate and Test é um processo repetitivo de tentativa-erro, onde um crítico analisa e escolhe os artefactos que melhor apresentam o resultado pretendido.

Search

Search é um método que consiste na procura de um artefacto num vasto espaço de resultados. Utiliza parâmetros de pesquisa de forma a encontrar o artefacto que satisfaça esses requisitos numa população de resultados já criados.

"The basic metaphor is that of design as a search process: a good enough solution to the design problem exists within some space of solutions, and if we keep iterating and tweaking one or many possible solutions, keeping those changes which make the solution(s) better and discarding those that are harmful, we will eventually arrive at the desired solution" (Procedural Content Generation in Games, Chapter 2, p. 17, para. 1).

Exemplo de algoritmos: *Hill Climbing* e *Genetic Algorithms* são utilizados de forma a otimizar o processo de pesquisa num vasto espaço de resultados.

Constraint Solving

Constraint Solving é um método que descreve e restringe o espaço de resultados de forma a encontrar um parâmetro válido. A partir de um problema e um conjunto de parâmetros, o método subtrativo consegue restringir o espaço de resultados.

"They are what you use when you have a lot of hard constraints, a lot of flexible and complex structure, but you don't know how to build out the structure in a way that will be sure to solve your constraints" (Compton, 2016).

Exemplo de algoritmos: Inverse Kinematics (IK-solving) e Wave Function Collapse (Gumin, 2017) são utilizados de forma a delimitar/restringir o espaço possível de resultados.

Implementar um Gerador Procedimental

Antes de começar a implementar um gerador procedimental é necessário avaliar a sua viabilidade.

A identificação do problema é uma importante fase na implementação de um gerador procedimental. Nem sempre é necessário desenvolver geradores. É essencial determinar se existe uma grande repetição de tarefas ou uma grande dimensão que não é viável por métodos tradicionais.
Depois de identificar o problema, segue-se a fase do "Proof of Concept" onde se resolve o problema num espaço relativamente curto de tempo e de uma forma mais simples possível, dividindo o problema em partes mais pequenas. É uma fase de prototipagem, de teste, de análise e de verificação de resultados. Nesta fase determina-se o objetivo do gerador procedimental. É uma fase decisiva que vai determinar o avanço do projeto, utilizando geradores procedimentais, e quais as ferramentas e os métodos a utilizar.

Por fim, a fase de implementação. Tendo em conta os processos anteriores, o gerador é aperfeiçoado, disponibilizando parâmetros de controlo e uma boa organização de funções.
Ao passar por todas estas fases e respondendo a estas questões antes de iniciar a implementação do gerador procedimental podem poupar-se muitas horas de trabalho.

Relação entre Utilizador e o Conteúdo

No caso do conteúdo digital ser criado por um gerador procedimental Online, como videojogos ou aplicações interativas, onde cada utilizador vai ter um resultado diferente, é essencial permitir e contribuir para que os utilizadores criem uma relação de propriedade com o conteúdo que é gerado.

"Player stories always be more powerful than scripted stories" (Wright, 2005) esta referência da apresentação "The Future of Content", reflete bem a importância das experiências únicas de cada utilizador, sendo por isso fundamental disponibilizar ferramentas de partilha do conteúdo gerado pelo utilizador com outros utilizadores.

Características das Técnicas Procedimentais

A implementação de técnicas procedimentais apresenta muitas vantagens e particularidades. Assim, vou utilizar a apresentação “Procedurally Crafting Manhattan for Marvel 's Spider-Man” (Santiago, 2019) como forma de melhor exemplificar as potencialidades destas técnicas e as suas particularidades.

Esta apresentação refere a utilização de um sistema procedimental para a geração do mapa de Manhattan. Este é um gerador offline, utilizado como ferramenta de desenvolvimento para utilização num jogo Open World. O mapa tem a dimensão de 6 por 3 km, dividido em 9 distritos, que incorpora um sistema de terreno, construções, trânsito, entre outros.

A utilização de métodos tradicionais para a criação de um mapa desta dimensão e complexidade seria impensável ou necessitaria de um grande período de desenvolvimento e de uma grande equipa.

Um gerador procedimental proporciona muitas características úteis que mudam por completo a forma como criamos conteúdo digital, como por exemplo a abstração, a flexibilidade e a versatilidade.

A abstração permite numa dissociação de um modelo específico, não necessitando de armazenar explicitamente todos os detalhes complexos de uma cena ou sequência. Em alternativa, utiliza-se uma função ou algoritmo e avalia-se esse procedimento quando e onde é necessário «content on demand». Isto permite criar modelos e texturas com uma resolução adaptável, de acordo com a resolução necessária.

Este nível de abstração permite uma reutilização deste gerador para outros objetivos semelhantes, como a geração de uma cidade diferente.

A flexibilidade permite uma grande adaptabilidade e adequação às necessidades. Como é referido na apresentação, o gerador foi utilizado o mais cedo possível no processo de criação do jogo, tendo em conta as diferentes fases de produção e eventuais adições futuras.

A versatilidade permite que um gerador procedimental execute diferentes tarefas. Neste caso, o gerador procedimental desempenha diferentes funções de geração do terreno, das ruas, dos edifícios, do trânsito, entre outras.

A utilização destas técnicas procedimentais no referido projeto permitiu:

- criação de muito conteúdo de jogo;
- consistência de resultados;
- remoção de tarefas repetitivas do processo de desenvolvimento;
- rápido processo de desenvolvimento, alocação de ferramentas para futuros projetos;
- alocação de custos e recursos.

Contudo, há fatores a ter em consideração no processo de desenvolvimento:

- longo processo de aprendizagem;
- perceção de perda de controlo;
- ser novo e diferente.

Que opção metodológica fez sentido prosseguir?

No desenvolvimento do Estado da Arte foi necessário consultar muitos artigos, apresentações e informações que, por vezes, podem apresentar diferentes terminologias para os mesmos conceitos, o que torna mais difícil o processo de seleção de informação.
No entanto, é uma área com uma grande quantidade de informação disponível online e apenas referencio aquilo que considero ser essencial, no que consiste a técnicas procedimentais. Apresento as origens, as definições, os tipos, as ferramentas, os métodos e as características das técnicas procedimentais.
De forma a prosseguir a investigação houve que se adaptar a metodologia quanto à natureza aplicada de forma qualitativa, exploratória, pesquisa e observação.

Conclusão

As técnicas procedimentais são cada vez mais utilizadas e estão a mudar a forma com criamos conteúdo digital.

Há fatores importantes a ter em consideração com o desenvolvimento desta tecnologia, nomeadamente:

- "Break down problems"
- Integrar em *workflows* existentes
- Facilitar e automatizar o trabalho

As técnicas procedimentais têm cada vez mais utilização, indo além da área dos videojogos e dos efeitos visuais. Atualmente, áreas como a arte e o design procedimental têm despertado cada vez mais atenção, mudando a forma como desenvolvemos o mobiliário, a arquitetura, a joalharia e os mais diversos objetos do quotidiano.

Com esta informação mais facilmente se consegue perceber e pôr em prática a geração procedimental, nomeadamente a geração de modelos 3D.

Métodos e Materiais

Com este projeto pretendeu-se criar uma ferramenta que automatiza a criação de modelos tridimensionais (3D) e demonstra as potencialidades de utilização de técnicas procedimentais no âmbito na computação gráfica, para qualquer pessoa que pretenda desenvolver e conhecer como automatizar o processo de desenvolvimento de conteúdo digital.

O processo de desenvolvimento do projeto foi dividido em três fases de execução: uma primeira fase de investigação para o desenvolvimento do estado da arte que decorreu até a 28 de fevereiro, a segunda fase de *proof of concept* e desenvolvimento prático até 31 de maio, por último, a terceira fase, de exportação dos dados, renderização e análise de resultados até 31 de julho.

Com base em toda a investigação feita no decorrer do estado da arte, escolhi abordar a parte prática de forma qualitativa, de natureza aplicada, com objetivo exploratório e procedimento experimental. Pretendi demonstrar a aplicação de técnicas procedimentais utilizando exemplos práticos e as ferramentas mais adequadas nesta área.

Desta forma, a parte prática foi dividida em três diferentes exemplos de geradores procedimentais offline, utilizando cada um diferentes métodos generativos. O primeiro é um gerador de rochas, o segundo é um gerador de árvores, e por último, um gerador de terreno.

A escolha de diferentes exemplos práticos permite explorar múltiplas abordagens, técnicas e aplicações de diferentes métodos generativos, enriquecendo o processo exploratório e diversificando os resultados alcançados.

Quanto às ferramentas usadas durante a parte prática, optei pela utilização do Houdini como a ferramenta principal para desenvolver todos os geradores procedimentais offline, Blender como ferramenta de renderização e DaVinci Resolve como ferramenta de correção de cor e exportação dos resultados obtidos. A escolha destas ferramentas foi feita com base no ambiente de desenvolvimento que utilizo (linux), e por já ter experiência de utilização.

Desenvolvimento

De forma a pôr em prática as potencialidades das técnicas procedimentais vou focar-me na criação de modelos 3D, mais especificamente em geradores procedimentais offline.

O desenvolvimento foi dividido em três partes que apresentam três diferentes métodos aditivos de geração procedimental: paramétricos, grammars e simulação.
O primeiro é um gerador de rochas, o segundo um gerador de árvores, e por último um gerador de terreno. Cada um destes geradores é dividido em diferentes etapas de implementação, como por exemplo a geometria, a textura, exportação ou a renderização do modelo 3D.
A implementação destes diferentes modelos procedimentais permite pôr em prática o que investiguei no estado da arte que permitiu a aprendizagem de diferentes ferramentas e técnicas.
Quanto às ferramentas, utilizei como base para todos os geradores o software Houdini, Blender para a renderização final e DaVinci Resolve para a composição final dos resultados obtidos.
Cada uma das diferentes etapas de desenvolvimento são abordadas de uma forma superficial, para mais facilmente compreender o funcionamento e lógica de implementação.

Parte 1 – Geração de Rochas

O gerador de rochas foi dividido em cinco etapas, que refletem as diferentes fases de desenvolvimento do gerador: a geometria, a otimização, a texturização, a automatização, a exportação e a renderização. O objetivo deste gerador é criar uma grande quantidade de rochas diferentes e exportar esses dados para que possam ser utilizados noutro software 3D ou game engine. Todo o processo de geração foi criado no Houdini.

Geometria

A primeira fase de desenvolvimento é a geração da geometria que vai dar forma à rocha. A geometria vai ser gerada com base numa esfera que vai passar por múltiplas distorções geométricas, utilizando diferentes parâmetros de ruído.

O ruído utilizado para deformar a esfera inicial que contribui para maior semelhança com uma formação rochosa é Worley Celluar.

Figura 1 - Ruído Worley Celluar com aumento gradual de fractal: 0%, 20%, 50%, 80% e 100%.

O ruído Worley Celluar, que se encontra nas imagens acima representado, é utilizado em diversas fases de distorção da geometria, começando por uma ordem de complexidade mais simples e progressivamente adicionando mais detalhes.

Utilizando o poder dos métodos aditivos paramétricos conseguimos atingir uma grande diversidade de resultados com o mesmo ruído, alterando apenas a frequência, o período, a amplitude e os valores mínimos e máximos e assim atingir o resultado pretendido.

Figura 2 - Diferentes fases de distorção da geometria.

As imagens acima demonstram as diferentes fases de distorção da geometria até ao resultado final.

Otimização

Depois de termos gerado a geometria é necessário otimizar o processo de geração. Para isso, foram criadas três diferentes geometrias para cada rocha, uma geometria proxy, uma geometria de alta resolução e uma geometria de baixa resolução.

Figura 3 - Geometrias da rocha: proxy, alta resolução e baixa resolução.

O proxy funciona como uma geometria que é gerada de forma mais rápida, comparando com a original (alta resolução), e apresenta uma aproximação da geometria da rocha, permitindo ajustar os valores de ruído e ter mais rapidamente um resultado final. Esta geometria apenas é utilizada no processo de desenvolvimento do gerador.

A geometria de alta resolução é uma geometria com elevado número de polígonos, é a geometria que demora mais tempo a ser criada e que tem mais detalhe. Nesta geometria terão origem as texturas e a geometria de baixa resolução.

A geometria de baixa resolução é uma geometria com baixo número de polígonos com mapeamento UV que é utilizado pelas texturas. Esta geometria vai ser utilizada para exportar para outras aplicações em conjunto com as texturas.

Texturização

O passo seguinte é a geração de texturas para cada geometria de rocha criada. Dependendo das necessidades e objetivo final de aplicação deste modelo, pode ser escolhida uma resolução diferente ou diferentes texturas. Neste caso, gerei texturas 4096 por 4096 (4K): uma textura para a cor principal (base color), uma textura para rugosidade (roughness) e uma textura para transferir os detalhes da geometria de alta resolução para a geometria de baixa resolução (displacement).

Figura 4 - Texturas da rocha: *base color*, *roughness* e *displacement.*

Para gerar as texturas para a cor principal e para a rugosidade é necessário a criação de um shader. Este shader é gerado combinando Worley Celluar e Perlin com diferentes valores de amplitude, de frequência e diferentes fractals, permitindo a criação de uma textura complexa, com semelhanças de uma rocha.

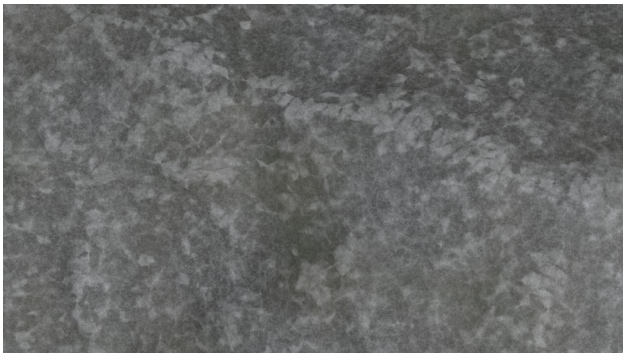

Figura 5 - Renderização do shader procedimental da rocha.

Depois de o shader criado, é necessário exportar utilizando o mapeamento UV da geometria de baixa resolução. Este processo tem o nome de texture baking onde se utiliza a geometria de alta resolução para gerar o shader e exporta-se as texturas para a geometria de baixa resolução.

Automatizar

O processo de geração da rocha acaba aqui. No entanto, se quisesse gerar outra rocha teria de alterar os parâmetros da geração da geometria e teria de voltar a gerar as texturas correspondentes e exportar novamente. Este gerador apenas consegue gerar uma rocha de cada vez.

Em cima deste gerador de rochas podemos criar outro gerador que vai automatizar o processo de geração de rochas, onde o utilizador apenas necessita de dizer quantas rochas pretende gerar e o gerador vai criar e exportar todas as rochas, cada uma com diferentes valores de ruído. No Houdini é possível criar utilizando Task Operator (TOP Network). Esta ferramenta permite criar Procedural Dependency Graph (PDG) que descreve quais os itens de trabalho a serem concluídos e as suas dependências e, assim, automatizar todo o processo de geração.

Exportar

O gerador exporta os modelos de rochas em dois formatos de ficheiros, em OBJ e em USD.

O formato OBJ é um padrão de armazenamento de modelos 3D e está presente nas mais diversas ferramentas 3D como por exemplo Blender.

O formato USD (Universal Scene Description) é um formato universal, criado pela Pixar, que permite armazenar uma grande diversidade de dados, desde geometrias, shaders, luzes, animações, físicas, entre outros. Este formato tem vindo a expandir em toda a indústria 3D e pode ser utilizado em ferramentas como Blender, Houdini ou na plataforma Omniverse (NVIDIA).

Renderização

A renderização final dos resultados obtidos foi feita no Houdini, utilizando Karma como render engine, onde foram renderizadas 20 rochas diferentes.

A composição final e correção de cor foram feitas no DaVinci Resolve.

Figura 6 - Renderização da rocha: sem texturas, apenas com *displacement*, com *displacement* e *base color*.

Parte 2 – Geração de Árvores

A geração de árvores foi dividida em quatro etapas, que refletem as diferentes fases de desenvolvimento do gerador: a geometria, a otimização, a texturização e a renderização. O objetivo deste gerador é criar uma grande quantidade de árvores diferentes, utilizando L-Systems. Todo o processo de geração e renderização foi desenvolvido no Houdini.

Geometria

Para a geração da geometria de uma árvore foi necessário a decomposição em elementos mais simples. A geometria foi dividida em três partes: o tronco, os ramos e as folhas. Cada uma destas geometrias foi gerada de forma individual, permitindo um maior controlo do resultado final, e posteriormente agrupadas para a geração de uma árvore.

O tronco e os ramos foram criados utilizando grammars como método aditivo, mais especificamente L-Systems. Este método permite a criação de um conjunto de regras que vão determinar a morfologia da árvore, possibilitando a criação de geometria complexa a partir de algumas regras.

As folhas são geradas com base numa textura de entrada com fundo transparente. Cada uma destas folhas foram selecionadas de forma aleatória e posicionadas no extremo de cada ramo.

Figura 7 - Ramo da árvore sem e com folhas.

Quando o tronco é gerado são distribuídos pontos ao longo da sua superfície. Cada um desses pontos dá origem a um ramo diferente, com as suas respetivas folhas.

Figura 8 - Decomposição dos diferentes elementos da árvore: tronco, ramos e folhas.

Otimização

Devido à complexidade da estrutura de uma árvore, é essencial otimizar o processo de geração, para isso foi necessário reutilizar geometria já criada e recorrer ao método Instancing.

Em vez de gerar um novo ramo com as respetivas folhas cada vez que é gerado um novo tronco, são gerados e armazenados cerca de 50 ramos diferentes. Estes ramos são depois utilizados e reutilizados para os diferentes troncos. Isto permite acelerar o processo de geração da árvore, sem ter a perceção de que os ramos são os mesmos.

A utilização de Instancing também permite otimizar todo o processo de geração e de renderização. Este processo consiste na utilização de pontos que se referem a outros objetos, uma instanced geometry permite a poupança de memória e aceleração do gerador.

Texturização

Ao contrário do mapeamento UV, as texturas utilizadas deste gerador não são geradas de forma procedimental.

As texturas do tronco e ramos da árvore são criadas a partir da digitalização 3D (*photogrammetry*) de um tronco de uma árvore, mas pode ser utilizado qualquer outra textura de tronco que melhor se adeque ao resultado pretendido. A partir da digitalização 3D foi extraída uma textura para a cor principal (*base color*) e outro mapeamento das normais (*normal map*).

Figura 9 - Point cloud da digitalização 3D e respetivas texturas: *base color* e *normal map*.

Para a textura das folhas, apenas foi utilizada a cor principal, criada a partir do recorte de fotografias de diferentes folhas. Esta textura tem de corresponder à mesma utilizada para gerar a geometria da folha.

Figura 10 - Recorte das folhas utilizadas para geração da árvore amarela e vermelha.

Renderização

Foi feita a renderização de dois tipos de árvores, a amarela e a vermelha. Cada uma tem diferentes texturas e diferentes atributos que contribuem para uma morfologia distinta.

Dada a dificuldade de exportar cada uma das árvores geradas com as respetivas texturas, devido à utilização de Instancing, apenas foi possível renderizar no Houdini utilizando Mantra como render engine.

Foram geradas e renderizadas cerca de 100 árvores de cada tipo, amarela e vermelha.

A composição final e correção de cor foram feitas no DaVinci Resolve.

Figura 11 - Renderização da árvore amarela e vermelha.

Parte 3 – Geração de Terreno

O gerador de terreno foi dividido em quatro etapas, que refletem as diferentes fases de desenvolvimento do gerador: a geometria, a exportação, a texturização e a renderização. O objetivo deste gerador é criar um terreno com as respectivas texturas para depois poder ser utilizado uma game engine ou software 3D.

A geração da geometria do terreno foi desenvolvida no Houdini, a texturização e renderização no Blender.

Geometria

A geração de geometria foi desenvolvida no Houdini, utilizando as ferramentas Height Field que permitem a distorção da geometria, em conjunto com a simulação de erosão.

O processo decorre de forma gradual, começando pela distorção da geometria de menor resolução, com o objetivo de delimitar as formas de vão dar origem ao terreno.

Utilizei o ruído Worley com uma elevada amplitude e gradualmente adicionei mais detalhe, reduzindo a amplitude e utilizando Curl Noise e Chebyshev Worley. Esta fase teve como objetivo delimitar as formas base do terreno antes do processo de erosão.

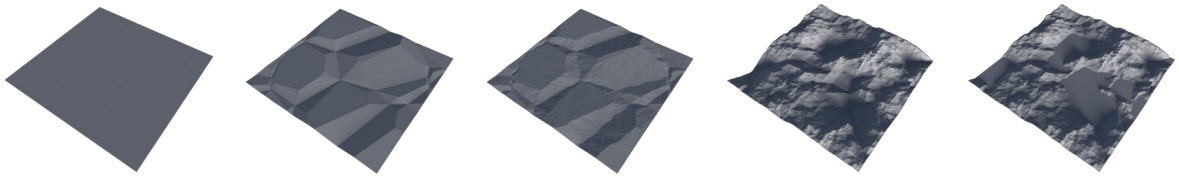

Figura 12 - Diferentes fases de distorção do terreno.

A utilização de métodos aditivos de simulação como a erosão contribui para um maior realismo no resultado final, simulando os efeitos de fenómenos naturais na paisagem.

Nesta fase, foram executadas duas erosões, tendo a segunda o dobro da resolução da primeira e, por último, tem uma leve distorção do terreno utilizando Curl Noise.

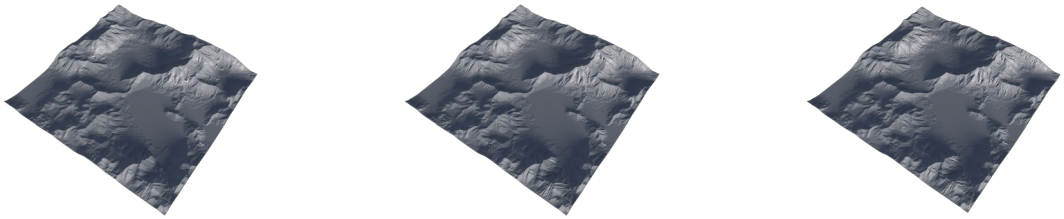

Figura 13 - Diferentes fases de erosão do terreno.

Exportar

Depois de ter a geometria gerada já posso exportar para outro software 3D. No entanto, há diversos atributos que são criados pelo Houdini ao longo do processo de erosão que facilitam o processo de texturização, ou mesmo de distribuição de vegetação. Para poder extrair estes atributos como height, bedrock, debris, sedimentos e water para o Blender tive de converter os atributos para vertex color. Só assim o Blender consegue interpretar os valores como atributos. Por fim, exportei toda a geometria do terreno e os atributos, utilizando o formato Alembic.

Texturização

Após exportação da geometria do terreno e respetivos atributos, inicia-se o processo de texturização do terreno no Blender.

Para gerar as texturas foi necessário a criação de um shader que vai utilizar os diferentes atributos como máscaras. Estas máscaras permitem uma maior facilidade na delimitação de cada superfície, utilizando diferentes texturas para cada superfície.

As texturas utilizadas como base para a criação do shader foram retiradas na plataforma Quixel Megascans, sendo escolhidas consoante o resultado final pretendido.

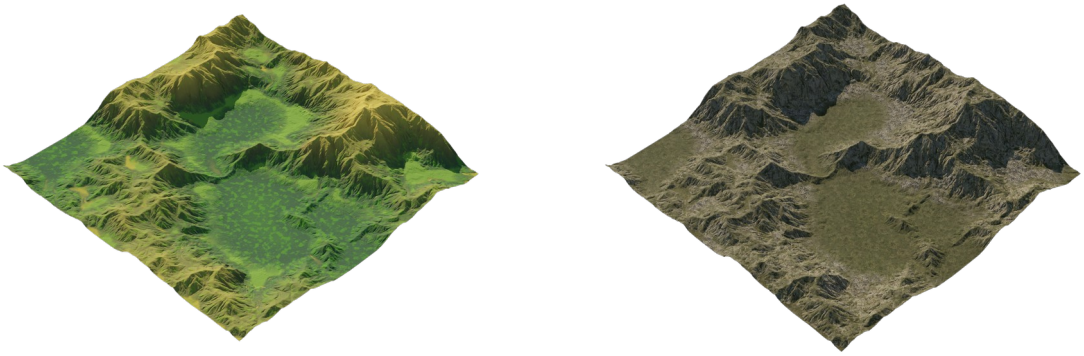

Figura 14 - Renderização do terreno com os respetivos atributos e com as texturas.

Renderização

A renderização foi feita no Blender, com recurso a render engine Cycles, tendo sido renderizados dois terrenos. A composição final e correção de cor foram feitas no DaVinci Resolve.

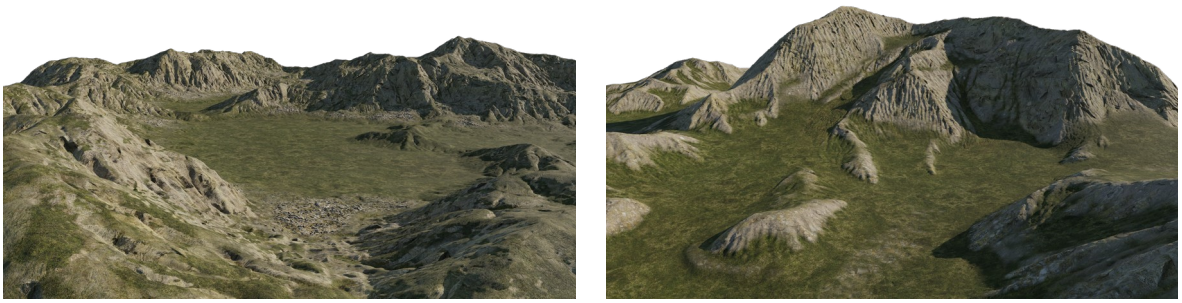

Figura 15 - Renderização de diferentes terrenos.

Conclusão

Com este projeto pretende-se criar uma ferramenta que automatiza a criação de modelos tridimensionais (3D) e demonstrar as potencialidades na utilização de técnicas e procedimentos no âmbito da computação gráfica.

O desenvolvimento da parte prática permitiu pôr à prova o conhecimento adquirido no durante a investigação do estado da arte. Pude, assim, aprender e desenvolver a geração procedimental, tendo como objetivo a diversificação de métodos generativos aditivos e subtrativos.

Com a realização dos três exemplos práticos foi possível verificar a importância e dificuldade da etapa de otimização, que inicialmente não estava prevista. Este processo de otimização requer um grande conhecimento das ferramentas e do seu funcionamento, tendo-se revelado uma etapa essencial, apesar de ser mais demorada. No entanto, sem esta etapa não seria possível uma boa responsividade do gerador de forma a acelerar o processo de criação.

A realização deste trabalho no âmbito no Projeto Global para a Licenciatura de Engenharia Multimédia, permitiu expandir o meu conhecimento na área de computação gráfica e ter uma melhor perceção do seu funcionamento. Possibilitou, também, conhecer novas ferramentas e técnicas que serão certamente úteis na minha vida profissional, dado que é uma área que pretendo aprofundar.

Em suma, com a realização deste projeto consegui explicar, demonstrar e aplicar as técnicas procedimentais, evidenciando as suas potencialidades na criação de conteúdo digital.

Referências Bibliográficas

Ebert, D. S., & Al, E. (2003). Texturing and modeling A procedural approach. Amsterdam [Etc.] Kaufmann Cop.

Rogue [Video game]. (1983) . A.I. Design

Elite [Video game]. (1984) . Acornsoft

RenderMan Documentation. (n.d.). Renderman Documentation. Wiki. https://rmanwiki.pixar.com/

Procedural Content Generation Wiki. (n.d.). Pcg.wikidot.com. [Wiki]. http://pcg.wikidot.com/what-pcg-is

Compton, K., Osborn, J. C., & Mateas, M. (2013, May). Generative methods. In The Fourth Procedural Content Generation in Games workshop, PCG (Vol. 1).

Quilez, I., & Jeremias, Shadertoy [Website]. https://www.shadertoy.com/

Polycount. (n.d.). Shaders [Wiki]. http://wiki.polycount.com/wiki/Shaders

Carpentier, G. (2011) Scape: 1. Rendering terrain. [Blog] https://www.decarpentier.nl/scape-render

Parberry, I. (2014). Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data, Journal of Computer Graphics Techniques, http://jcgt.org/published/0003/01/04/

KellerDev, (2020). Procedural Neighbourhood Generation in Houdini 18.0. [Video] https://youtu.be/5NNHSTXwmSQ

Solberg, T. (2021). Transforming animation with machine learning. Embark Studios. https://medium.com/embarkstudios/transforming-animation-with-machine-learning-27ac694590c

Rosen, D. (2014). An Indie Approach to Procedural Animation (Wolfire Games, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1020583

Hecker, C. (2007). How To Animate a Character You've Never Seen Before (definition six, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/759

Richter, J. (2020). Destructible Environments in 'Control': Lessons in Procedural Destruction (Remedy, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1026820

Wolverson, H. (2020) Procedural Map Generation Techniques, Roguelike Celebration [Video] https://youtu.be/TlLIOgWYVpI

Martens, C., & Cardona-Rivera, R. (2017). Procedural Generation of Cinematic Dialogues in 'Assassin's Creed Odyssey' (Ubisoft Quebec, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1024143

Paradis, F. (2019). Procedural Narrative Generation (North Carolina State University, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1025980

Podolny, S. (2015). If an Algorithm Wrote This, How Would You Even Know? [The New York Times]. https://www.nytimes.com/2015/03/08/opinion/sunday/if-an-algorithm-wrote-this-how-would-you-even-know.html

David Li, Blob Opera. Google Arts & Culture. Retrieved October 6, 2021, from https://artsandculture.google.com/experiment/blob-opera/AAHWrq360NcGbw

Grendel Games. (2018, June 1). Procedural Generation. https://grendelgames.com/procedural-generation/

Houdini [Software] SideFX. Website. https://www.sidefx.com/products/houdini/

Substance [Software] Adobe. Website. https://www.adobe.com/

Kruel, L. (2017). Introduction to Proceduralism (SideFX Software, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1024281

McKendrick, I. (2018). Working With Change (Hello Games, Ed.) [Everything Procedural Conference]. http://www.everythingprocedural.com/EPC2018.html

Minecraft [Video game]. (2011) . Mojang Studios

No Man’s Sky [Video game]. (2016) . Hello Games

Horizon Zero Dawn [Video game]. (2017) . Guerrilla Games

McKendrick, I. (2017). Continuous World Generation in 'No Man’s Sky' (Hello Games, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1024265/

Muijden, J. van. (2017). GPU-Based Run-Time Procedural Placement in 'Horizon: Zero Dawn' (Guerrilla Games, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1024120/

Tilt Brush [Software]. Google. Website. https://www.tiltbrush.com/

Introduction to Procedural Modeling, SideFX. [Website]. https://www.sidefx.com/learn/collections/introduction-to-procedural-modeling/

Schwind, M., Merkle, M. Design/VFX/Generative Art. FMX (2017) [Video]. https://player.vimeo.com/video/217206682?h=af61ad91c9

Blender [Software] Blender Foundation. Website. https://www.blender.org/

Everything Nodes. Blender Developer Wiki. [Wiki]. https://wiki.blender.org/wiki/Source/Nodes/EverythingNodes

Geometry Nodes. Blender Manual. [Doc]. https://docs.blender.org/manual/en/latest/modeling/geometry_nodes

Substance 3D Designer Documentation. Adobe. [Doc] https://substance3d.adobe.com/documentation/sddoc/substance-3d-designer-102400008.html

Dichelle, J., & Rood, L. (n.d.). Nodevember [Website]. https://nodevember.io/

Compton, K. (2016). So you want to build a generator [Blog Post]. GalaxyKate. https://galaxykate.com/blog/generator.html

Civilization V [Video game]. (2010) . Firaxis Games

Zhao, A., Xu, J., Konaković-Luković, M., Hughes, J., Spielberg, A., Rus, D., & Matusik, W. (2020). RoboGrammar: Graph Grammar for Terrain-Optimized Robot Design. ACM Trans. Graph, 39. https://doi.org/10.1145/3414685.3417831

Murray, S. (2017). Building Worlds Using Math(s) (Hello Games, Ed.) [Game Developers Conference]. https://www.gdcvault.com/play/1024514

Willmott, A. (2007). Fast Object Distribution (Electronic Arts, Ed.) [Fast Object Distribution]. Https://Www.cs.cmu.edu/~Ajw/S2007/0312-ObjectDistribution.pdf; Siggraph.

A. Adonaac, (2015). Procedural Dungeon Generation Algorithm. Game Developer. https://www.gamedeveloper.com/programming/procedural-dungeon-generation-algorithm

Gardner, Martin. 'Mathematical Games: The Fantastic Combinations of John Conway’s New Solitaire Game Life.' (Scientific American 229, Ed.) [Website]. http://www.ibiblio.org/lifepatterns/october1970.html

Green, D. (2016). Procedural content generation for C++ game development. Packt Pub.

Noor Shaker, Togelius, J., Nelson, M. J., & Springer International Publishing Ag. (2018). Procedural Content Generation in Games. Cham Springer International Publishing Springer. https://www.academia.edu/1578545/Procedural_content_generation_in_games

Gumin, M. 2017. Wave Function Collapse. GitHub.
https://github.com/mxgmn/WaveFunctionCollapse

Wright, W. (2017). The Future of Content (Stupid Fun Club, Ed.) [Game Developers Conference].
https://www.gdcvault.com/play/1019981

Santiago, D. (2019). Procedurally Crafting Manhattan for 'Marvel's Spider-Man' (Insomniac Games, Ed.) [Game Developers Conference]. https://gdcvault.com/play/1025765

Anexos