

# Geração Procedural de Estórias: o uso de NPCs procedurais para complementar o jogo

Emiliandro Firmino

Jameson Pagini

Marco Henrique

Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Brasil

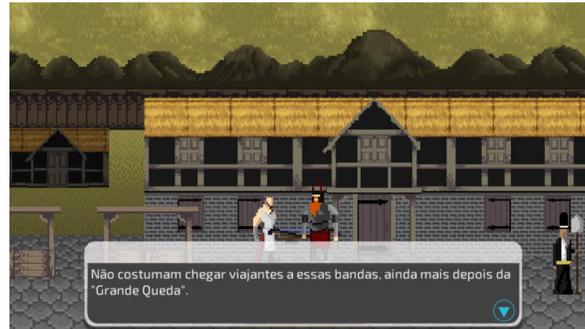


Figura 1: Exemplo de diálogo no jogo.

## RESUMO

No desenvolvimento de estórias para jogos, muito tempo acaba sendo gasto com personagens secundários que não afetam o rumo do enredo principal. Visando agilizar o processo de criação de jogos no mercado brasileiro, este artigo propõe o Corsário, uma inteligência artificial de fácil uso para *Unity*, que gera proceduralmente diversas informações para caracterizar *NPCs* que possam vir a interpor de inúmeras formas nas missões secundárias que o jogador irá receber. Dado o exposto, foi implementado um jogo onde essas “biografias” são utilizadas de diversas formas para enriquecer a experiência com o produto.

**Palavras-chave:** geração procedural, jogos digitais, enredos para jogos.

## 1 INTRODUÇÃO

Geração Procedural é a geração de conteúdo multimídia, através de algoritmos, a qual é usada em vários jogos e é de grande importância. O conteúdo gerado engloba, dentre outros, música, cenários e personagens para que a cada novo jogo seja uma experiência totalmente nova para o jogador. Essa ferramenta vem sendo usada por diversos desenvolvedores independentes, como uma forma de produzir mais conteúdo em menos tempo de desenvolvimento, para disputar com a grande variedade de jogos existentes e os serviços de *streaming* que vem oferecendo cada vez mais jogos AAA a custos baixíssimos.

Ademais, para garantir a qualidade do jogo, Trovato [1] atribui ao enredo e à sua complexidade a responsabilidade por prender o jogador. Logo, se faz necessário aos desenvolvedores implementarem personagens não jogáveis (*NPCs*) profundos e que transmitam ao jogador uma sensação de veracidade no que está sendo demonstrado.

Algumas questões surgem dessa situação, pensando nisso, é proposto nesse artigo o Corsário, que usa a geração procedural para escrever novos perfis de personagens secundários a cada

novo jogo, para auxiliar o desenvolvedor brasileiro a conciliar o tempo de desenvolvimento com a criação de personagens complexos para complementar o jogo.

## 2 TRABALHOS RELACIONADOS

Muitos autores têm estudado o uso da Geração Procedural em jogos, mas dentre eles foram escolhidos quatro, Gabriele Trovato, Soren John, Pieter Spronck e Larry Lebron. Os três primeiros tratam de usar uma inteligência para criar um ecossistema de um jogo com base em máquinas para testar as possibilidades em ambientes distintos existentes no mundo de jogo, a cada rodada de testes o desenvolvedor pode observar a evolução de cada comunidade, no entanto os habitantes dessas comunidades foram programados para não possuir profundidade alguma. Já Larry Lebron propõe uma I.A. para uso em pré-produções, chamada *Personage*, que é um sistema que promete criar e dar vida a um personagem através de fichas de interesses e, simultaneamente, biografando a vida de personagens de forma procedural até o presente tempo do enredo para assistir no desenvolvimento de jogos, livros e seriados.

## 3 PROPOSTA

Tendo a inteligência *Personage* [2] como ponto de partida, foi desenvolvido para ser aplicado diretamente no jogo, através da filtragem de diversos bancos de dados para escrever em um arquivo, o perfil do personagem que vai ser imediatamente instanciado em algum lugar no mundo do jogo portando animações e informações próprias. Entre as informações presentes nos perfis gerados, há algumas que podem vir a influenciar o jogo, como qual missão aquele perfil irá passar para o protagonista, onde ocorrerá a missão e qual a recompensa pela mesma. Tal ferramenta pode ser usada de diversas formas. Com isso, o mercado brasileiro pode ter uma ferramenta em português para agilizar o desenvolvimento de conteúdo para

\*e-mail: ecdmf.tjd@gmail.com

jogos digitais. Visando facilitar a compreensão, alguns conceitos estão explicados abaixo:

- 1) **LitJson**: foi usada a biblioteca LitJson no projeto Unity para que o Corsário crie as biografias de cada personagem em um arquivo leve e de fácil leitura.
- 2) **Unity**: a Unity Engine, por ser uma ferramenta de desenvolvimento de jogos completa e grátis, foi escolhida para ser a ferramenta de desenvolvimento desse jogo. Atualmente (2016) em sua quinta versão, ela permite desenvolver jogos 3D e 2D com noções simples de Programação Orientada a Objetos.
- 3) **NPCs**: é uma expressão da língua inglesa, Non-Player Character, que literalmente traduzida seria “Personagem Não Jogável”, denominação dada aos personagens dentro dos jogos que não são controlados pelos jogadores, mas que estão no enredo.

O Corsário tem duas fases de atuação: uma na Escrita de Perfil e outra na Leitura de Perfis. Ambas serão explicadas a seguir.

### 3.1 Escrita do Perfil

Ao início do desenvolvimento desses personagens foi escrita uma classe chamada *NPC*, que contém apenas variáveis e um construtor para ser chamado no *escritor.cs*. Através de inteiros e *arrays*, três profissões são escolhidas de forma que uma não venha a se repetir, com esse dado são montadas as informações básicas em *strings* e usadas como os valores do construtor *NPC()*, o objeto é salvo e mapeado com uma variável *JsonData* (pertencente à biblioteca *LitJson*) para ser escrito através do método *File.WriteAllText()*, no caminho endereçado pela biblioteca *System.IO*, assim sendo salvos em formato *Jsons* para acesso futuro.

```
using UnityEngine;
using System.Collections;

public class NPCs
{
    public string Nome, Sobrenome, LugarDeOrigem,
    Profissao, Crenca, Item;
    public NPCs(string nome, string sobrenome,
    string lugarDeOrigem, string profissao,
    string crenca, string item)
    {
        this.Nome = nome;
        this.Sobrenome = sobrenome;
        this.LugarDeOrigem = lugarDeOrigem;
        this.Profissao = profissao;
        this.Crenca = crenca;
        this.Item = item;
    }
}
```

Figura 2: Foto da classe chamada de NPC.

### 3.2 Banco de Dados em Array

As informações básicas de baixo grau de complexidade, ao qual o desenvolvedor deve preencher com nomes masculinos, nomes femininos e sobrenomes.



Figura 3: Fluxo de escolhas do perfil.

As questões mais voltadas à temática do jogo foram classificadas como sendo de médio grau, local de origem, local de item, informações sobre a cultura e o mundo do jogo em geral, que no final acabam sendo os *arrays* que mudam de jogo para jogo. Em alto grau ficam as questões da psicose humana, as índoles, a personalidade, e os aspectos físicos entram nesses *arrays* curtos, porém de suma importância para os diálogos.

#### 3.2.1 Índoles

Com intuito de aprofundar os perfis procedurais, o passo seguinte é a adição de suas opiniões sobre o mundo em que o jogador irá desbravar. Baseado em pesquisas sobre as teorias sobre os falsos ídolos de Bacon, onde, resumindo e omitindo informações, um ser humano possui um conhecimento básico sobre o mundo, sobre sua cultura, sobre sua crença e sobre si mesmo. Ademais, somado com conceitos do jogo de mesa *Dungeons & Dragons*, permitiu criar três afiliações distintas que definem a repercussão da missão - se ela é útil, inútil ou uma tentativa de mata-lo. Para definir esses padrões, por ser pouco aleatório foi adicionado como *enum* na classe *NPC*.

#### 3.2.2 Personalidades

```
public enum Personalidade {
    Honesto, // Fala normalmente
    Mentiroso, // Fala confusamente
    Tímido, // Fala pouco
    Excluído // Evita falar
}
```

Figura 4: Lista de possíveis personalidades

#### 3.2.3 Crença

Baseada na história do jogo, aspecto incorporado apenas para adicionar profundidade ao enredo.

#### 3.2.4 Afiliação

```
public enum Afiliacoes {
    Good, // Quests uteis
    Neutral, // Quests inúteis
    Evil, // Quests suicidas
}
```

Figura 5: Lista de possíveis afiliações.

### 3.2.5 Profissão

Baseada nas artes do jogo, aspecto adicionado apenas para instanciar o *NPC* no mundo do jogo contendo animação própria.

### 3.2.6 Leitura

Um terceiro *script*, *leitor.cs*, é anexado a um objeto dentro de jogo que possui um componente da *Unity*, o *UnityEngine.Animator*, também conhecido como biblioteca de animações do objeto, e um *array* contendo o nome de cada animação. Inicialmente o leitor, através da biblioteca *System.IO*, lê o perfil instanciado e com base na profissão define sua animação *default*. Ademais, suas variáveis são armazenadas em terceiros para que as informações necessárias sejam expostas nos diálogos e descrições.

```
{
  "Nome": "Leonor",
  "Sobrenome": "Zoras",
  "LugarDeOrigem": "London",
  "Profissao": "Bardo",
  "Crenca": "o deus das festividades",
  "Item": "Violao"
}
```

Figura 6: Perfil simples gerado em Json.

### 3.2.7 Testes

Os primeiros testes consistiam em gerar quatro perfis com base em *arrays*, com oito profissões e dez nomes masculinos, femininos e sobrenomes, sendo instanciados num total de 18 vezes para criar um banco de 72 perfis para a análise. Entre eles, 4% possuíam “*null*” como informação, deixando duas partidas impossíveis de serem completadas. A primeira correção veio com a diminuição do escopo do projeto para três perfis por jogo, assim, nos 72 perfis seguintes do segundo teste foi obtido 0% resultados “*null*”. Outras correções vieram apenas em aumentar a quantidade de itens dos *arrays* usados como banco de dados.



Figura 7: NPC gerado em teste.

### 3.2.8 Observação

Em caso de múltipla geração, é necessário adicionar novas linhas de código para controle de tipos de personagens e locais instanciados nos respectivos “leitura” e “escritor”.

## 4 JOGO DESENVOLVIDO

Seguindo as regras da *Adventure Jam*, um evento da *web* internacional de duas semanas voltado para a criação de jogos sobre contar uma história a respeito de pessoas, mundos ou objetos, *Restless* tem como objetivo principal concluir missões em troca de mais informações sobre o enredo. Desse modo o jogador irá encontrar *NPCs* gerados proceduralmente necessitando de ajuda, assim, passando as missões para o jogador executar, este normalmente encontrará alguns obstáculos e desafios projetados, entre eles, vencer os monstros que percorrem a fase e pegar o item em poder dos vilões, pois ao devolver o item, mais diálogos acabam sendo desbloqueados.

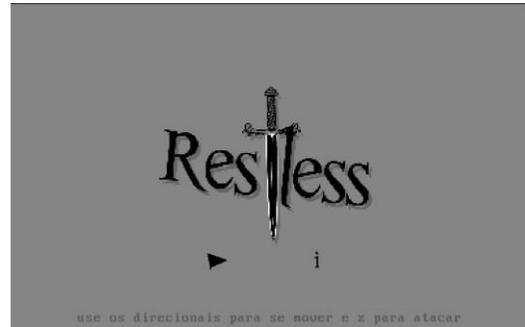


Figura 8: Menu inicial do jogo desenvolvido.

### 4.1.1 Fungus

Durante a leitura dos perfis foi decidido usar o *Fungus* que é uma extensão do *Unity*, voltada para criação dos diálogos.

### 4.1.2 Uso do Corsário

Durante o desenvolvimento, se tornou perceptivo que cada jogo possui uma forma específica de leitura e escrita. No caso do jogo desenvolvido, a profissão foi o primeiro a ser lido para em seguida definir o sexo (nome), isso devido a uma restrição nas imagens de cada personagem. Exemplos: a profissão “ladrão” tem como ícone uma mulher, a profissão “ferreiro” tem como ícone um homem, entre outros. Essa peculiaridade precisa de muita atenção quando utilizar-se o Corsário com qualquer jogo.

## 5 CONCLUSÃO

Neste trabalho foi apresentado um modelo de Inteligência Artificial para a criação procedural de personagens secundários, o Corsário. Até o momento foi implementado informações básicas como: Nome, Sobrenome, Lugar de Origem, Crença, Afiliação, Personalidade e Profissão.

O modelo se mostrou eficiente, pois enriqueceu os personagens existentes. Para que o escritor de *NPCs* tenha seu objetivo alcançado falta a implementação de informações mais complexas, tornando mais realistas os perfis gerados, no entanto tal implementação acaba por ser complicado, afinal, é muito particular do enredo de cada jogo. Com os resultados obtidos até agora, as histórias se tornam mais enriquecidas, porém não mais críveis.

Logo, pôde ser afirmado que jogos futuros gastarão menos tempo de desenvolvimento aprofundando personagens secundários com o Corsário, permitindo os desenvolvedores trabalharem em outras áreas do jogo, resultando na confecção de um produto mais viável, completo e, principalmente, um bom nível de profundidade para os personagens.

## 6 TRABALHOS FUTUROS

Devido ao pouco tempo de produção, é planejado o aumento do escopo e facilitação do uso do Corsário para jogos com personagens multiculturais, multifário e multilateral. Dessa forma, criando uma maior variedade de diálogos e missões, possibilitando jogos com múltiplos finais de acordo com a relação que o jogador tem com esses personagens.

## 7 AGRADECIMENTOS

Os autores gostariam de agradecer aos seus professores Adriano Gil, André Machado e seu orientador Dr. Cleto Leal, além da Universidade do Estado do Amazonas por proporcionar o curso de oferta especial de Tecnólogo em Jogos Digitais, do qual são bolsistas. Por fim, agradecer aos seus amigos Pedro Victor Ferreira e Silvio Lasmar pela parceria.

## REFERÊNCIAS

- [1] G. Trovato. S. Johnson. P. Spronck. Procedural Generated History: building a game ecosystem through autoplay. 10<sup>th</sup> International Conference on the Foundations of Digital Games. June 2015.
- [2] L. LeBron. Procedural Character Generation for Narrative Games. UC Santa Cruz. December 2012.
- [3] L. Pulsipher. An Introduction to Dungeons & Dragons, Part V: Characterization and Alignment. White Dwarf. Games Workshop. November 1981.
- [4] A. Kope. C. Rose. M. Katchabaw. Modeling Autobiographical Memory for Believable Agents. Western University. December 2013.