ISSN: 2675-1828

ARTIGO

Um algoritmo evolutivo para geração procedural de conteúdo para o jogo Super Mario Bros

An evolutionary algorithm for procedural content generation for the game Super Mario Bros

Mayara Beatriz Maia Silva[1]
Kleber de Oliveira Andrade[2]

Resumo: *Geração procedural de conteúdo (PCG) é um método de criação automatizada de conteúdo digital. Um dos desafios da área é gerar uma variedade de conteúdo, com custo computacional baixo e com garantia de validade. O presente trabalho aborda a geração procedural de níveis típicos do jogo Super Mario Bros (SMB). Este projeto considera a representação de um nível de SMB, por meio de um vetor de fatias verticais, assim, é possível criar um conjunto de padrões e utilizá-los na geração do mapa. Portanto a proposta é utilizar Algoritmos Evolutivos (AEs) combinados com os padrões (conjunto de fatias de uma fase real do jogo) para gerar os níveis do SMB. Os resultados preliminares demonstraram que, apesar da utilização de poucos padrões, os experimentos realizados convergiram para níveis viáveis.*

Palavras-Chave: *Algoritmos Evolutivos; Geração Procedural de Conteúdos; Inteligência Artificial.*

Abstract. *Procedural content generation (PCG) is a method of automated creation of digital content. One of the challenges in the area is to generate a variety of content, with low computational cost and guaranteed validity. The present paper addresses the procedural generation of typical levels of Super Mario Bros (SMB) game. This project considers the representation of a SMB level, by using a vector of vertical slices; accordingly, it is possible to create a set of patterns and use them in the generation of the map. Therefore, the aim is to use Evolutionary Algorithms (AEs) combined with the patterns (set of slices from a real game phase) to generate the SMB levels. The preliminary results showed that, despite the use of few standards, the conducted experiments converged to feasible levels.*

Keywords: *Artificial intelligence; Evolutionary Algorithms; Procedural Content Generation.*

[1] Faculdade de Tecnologia de Americana. E-mail: mayarabeatrizmaia@outlook.com
[2] Faculdade de Tecnologia de Americana. E-mail: kleber.andrade@fatec.sp.gov.br

1. Introdução

Os jogos digitais estão em constante inovação. Por conta do mercado competitivo e da necessidade de adaptação, os jogos tornaram-se cada vez mais complexos. Embora muito de seu campo seja acadêmico, os jogos tornaram-se cada vez mais específicos em suas habilidades e, por conta disto, surgiu uma necessidade de adaptação.

Por meio de Algoritmos Genéticos em jogos, é possível apresentar resultados eficientes em melhorias de jogos (KIM; KIM, 2020), gerar fases em diferentes níveis, gerar um algoritmo único para complexidade em fases de jogos de quebra-cabeça (FERREIRA, 2015), e apresentar conhecimento e planejamento de masmorras que dificultam o desempenho do jogador durante sua jogabilidade (VIANA, 2019).

O objetivo do presente artigo, portanto, é analisar a metodologia dos Algoritmos Genéticos sob uma perspectiva dentro dos jogos digitais. Utilizou-se como referência a geração de níveis aplicadas no jogo Mario (1981 – presente), conforme criado em *"Patterns as Objectives for Level Generation"* (DAHLSKOG, S., TOGELIUS, J., 2013.)

O trabalho está estruturado em referenciais teóricos para a maior compreensão do uso de AGs em jogos digitais, detalhes sobre Algoritmos Genéticos e qual sua aplicação neste projeto, materiais e métodos utilizados para esta aplicação e objetivos propostos, dois diferentes experimentos para discutir resultados e, por fim, conclusões e possíveis aplicações futuras.

2. Referencial teórico

Esta seção apresenta os conceitos fundamentais para entendimento deste trabalho, tais como, o algoritmo evolutivo e suas aplicações em geração procedural de fases em jogos digitais.

2.1. Algoritmo Evolutivo

O Algoritmo Evolutivo (AE) é uma técnica de busca e otimização inspirada na evolução natural das espécies. Este algoritmo foi inicialmente desenvolvido por Holland (1975). Segundo Eiben e Smith (2015) o princípio básico de todo AE, é o mesmo:

- Geração aleatória de indivíduos (somente na primeira geração). Cada indivíduo ou solução também é conhecido como cromossomo ou genoma;
- Avaliação da população seguindo algum critério determinado por uma função que calcula a qualidade individual de cada cromossomo (aptidão ou *fitness*);
- Seleção de indivíduos para realizar reprodução (crossover) por meio do seu *fitness*;
- Imposição de variações aleatórias nos cromossomos dos indivíduos resultantes (mutações);
- Reinício do processo com os novos indivíduos (nova geração);

• Estes passos são repetidos até que um critério de parada preestabelecido seja atingido.

Ao final da aplicação do evolutivo, obtêm um ou mais indivíduos evoluídos, ou seja, soluções mais aptas ao ambiente.

Diversos elementos desse processo evolutivo são estocásticos: a seleção favorece indivíduos mais bem adaptados, mas existe também a possibilidade de outras soluções serem selecionadas. A recombinação dos indivíduos também é aleatória, assim como a mutação. Destaca-se que é possível adotar também a técnica de substituição de indivíduos baseadas no conceito do elitismo, evitando a perda de soluções de alta aptidão (EIBEN e SMITH, 2015; DE JONG, 2012).

2.2. Algoritmo Evolutivo para geração procedural de fases

Dahlskog e S., Togelius (2013) desenvolveram o uso de padrões como objetivo de evolução de fases. Nesta implementação, são representadas como um conjunto de fatias verticais do jogo original, buscando encontrar o número de padrões, criando níveis gerados e seus pontos fortes do método implementado. Mariño, Pereira e Toledo (2017) demonstram os conceitos de geração procedural de conteúdos aplicados ao uso de jogos digitais e diferentes fases, demonstrando como alternativa de redução de custos e produção de jogos, tornando os conteúdos mais automatizados e, portanto, mais fáceis de desenvolverem jogos com maiores níveis de complexidade.

Ferreira (2015) demonstra o uso de geração procedural em conteúdo, criando aplicações em jogos de quebra-cabeça com a mecânica do jogo Angry Birds. Uma representação de níveis em forma de indivíduos fora introduzida, permitindo que o algoritmo genético evolua com diferentes características, resultando em níveis imersivos de forma automatizada, sem a necessidade de uma produção manual.

Viana (2019) demonstra o uso de geração procedural de jogos em Exploração de Masmorras (como, por exemplo, o jogo Dungeon Crawler). Barreiras são elementos que impedem o progresso do jogador. Nesta demonstração, utilizaram-se algoritmos genéticos para criação de barreiras de forma automatizada sob múltiplas combinações. Os algoritmos resultaram em níveis viáveis com mecânicas de barreiras complexas, alguns em superação aos níveis desenvolvidos manualmente. Costa Junior e Lisboa Ramalho (2014) desenvolveram gerações procedurais de conteúdos sob o uso de algoritmos genéticos para níveis de jogos infinitos. Os algoritmos demonstraram uma eficiência de desenvolvimento e complexidade em comparação a desenvolvimentos manuais, sendo útil para um ambiente mais imersivo dentro do campo de jogos infinitos.

3. Materiais e Métodos

Esta seção apresenta as ferramentas utilizadas, o ambiente desenvolvido, como foi projetado o cromossomo, a função de avaliação e os operadores genéticos utilizados.

3.1. Super Mario Bros

Mario é uma franquia de mídia produzida e publicada pela Nintendo, empresa japonesa. Fora originalmente criada pelo designer Shigeru Miyamoto em 1981. Considera-se, atualmente, uma das franquias de jogos mais influentes de todos os tempos; especialmente por estar em grande relevância até os dias de hoje, mais de trinta anos depois de seu lançamento. A franquia já lançou mais de 200 jogos diferentes.

Em Super Mario Bros (SMB), o jogador assume o papel de Mario, o protagonista da série. O irmão mais novo de Mario, Luigi, é controlado pelo segundo jogador no modo multiplayer do jogo e assume o mesmo papel e funcionalidade na trama de Mario. O objetivo é correr pelo Reino do Cogumelo, sobreviver às forças do principal antagonista de Bowser e salvar a Princesa Toadstool. O jogo é um jogo de plataforma side-scrolling; o jogador se move do lado esquerdo da tela para o lado direito a fim de alcançar o mastro da bandeira no final de cada nível.

3.2. Cromossomo e População inicial

Uma fase do jogo SMB pode ser representada por meio de um vetor de fatias verticais. Cada fatia tem largura de 1 bloco e altura de 15 blocos. Dahlskog e Togelius (2013) utilizaram um vetor de 200 fatias de 24 tipos. Cada fatia representada por uma letra do alfabeto.

Este projeto irá utilizar uma solução parecida, porém utilizará somente 14 tipos de fatias verticais. Sendo assim, o cromossomo terá 200 genes e cada gene poderá ser representado por valores de [A – N]. A Figura 3 apresenta as fatias verticais definidas após observar a primeira fase de SMB.

Vale ressaltar que um cromossomo de tamanho 200, no qual, cada posição do vetor pode assumir 14 valores diferentes, isso cria um universo de solução de 14^{200} soluções.

Após definir o formato do cromossomo, é possível criar uma população inicial. Essa população será composta por indivíduos, com seus genes aleatórios dentro dos limites permitidos.

3.3. Função de avaliação

Um dos pontos importantes para o AE convergir, é criar uma boa função de avaliação. Como visto anteriormente, essa função avalia a proximidade de uma determinada solução com a solução ideal do problema desejado. Neste caso, será utilizado uma busca linear em cada indivíduo por padrões pré-definidos, no qual, cada padrão terá um valor quantitativo. Quanto mais padrões existirem na solução, maior será o *fitness* do indivíduo. Sequências com jogabilidade impossível terão valores negativos.

A Tabela 1 exibe os padrões e seus respectivos pontos utilizados para avaliação de um cromossomo. Os valores foram definidos de forma empírica imaginando o que seria mais interessante para a fase. Para exemplificar estes padrões, observe a Figura 4 no canto direito, existe o padrão CDEDC (Figura 3).

Tabela 1. Padrões definidos para a função de avaliação.

Padrões	Ponto(s)	Padrões	Ponto(s)
AA	1	FGHI	2
BB	1	IHGF	2
CDEDC	3	JEEJ	2
DBLBD	3	AAAAA	-2

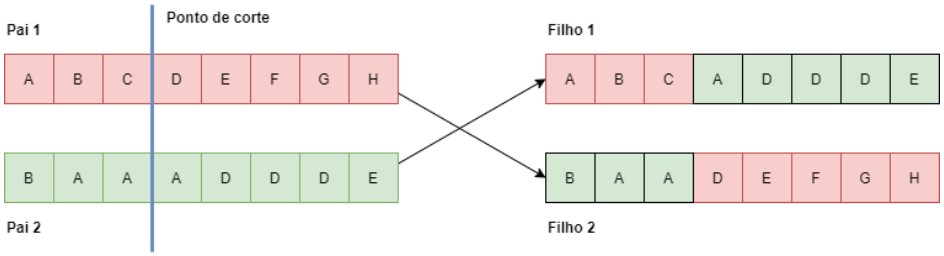
Fonte: Elaborado pelos autores (2020).

3.3. Operadores Codificados

O método de seleção utilizado é o por Torneio (EIBEN e SMITH, 2015), no qual, 2 indivíduos são escolhidos aleatoriamente e o que obtiver o melhor *fitness* é selecionado para reproduzir. Esse processo é realizado para selecionar 2 pais, e assim, recombinar suas informações.

Holland foi um dos precursores na análise de operadores de recombinação e propôs o que hoje é conhecido como recombinação de um ponto (EIBEN e SMITH, 2015). Para a aplicação deste operador, são selecionados dois indivíduos (pais) e a partir de seus cromossomos são gerados dois novos indivíduos (filhos). Para gerar os filhos, seleciona-se um mesmo ponto de corte aleatoriamente nos cromossomos dos pais, e os segmentos de cromossomo criados a partir do ponto de corte são trocados. Considere, por exemplo, dois indivíduos “ABCDEFGH” e “BAAADDDDE” e que o ponto de corte escolhido aleatoriamente tenha sido 3, então os filhos gerados seriam: “ABCADDDE” e “BAADEFGH” a Figura 4 retrata este exemplo.

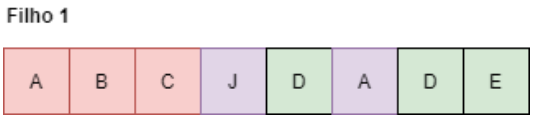
Figura 4. Recombinação uniforme aplicada a um cromossomo.



Fonte: Elaborado pelos autores (2020).

Logo após a recombinação, cada filho gerado tem 1% de chance de ter a informação de nenhum ou muitos genes resetados - gera-se um novo valor aleatório dentro dos limites permitidos (EIBEN e SMITH, 2015). A Figura 5 utiliza o filho 1 gerado no exemplo da Figura 4, para ilustrar a mutação no gene 4 e 6.

Figura 5. Mutação aleatória aplicada a um cromossomo.



Fonte: Elaborado pelos autores (2020).

Para finalizar, é possível aplicar elitismo para não perder as melhores soluções de uma geração. Além disso, como critério de parada geralmente é definido uma quantidade de gerações – este projeto utilizará 1000 gerações.

4. Resultados

Alguns experimentos foram realizados para comprovar que o AE projetado, é capaz de criar cenários para jogos de plataforma no estilo SMB. Para isso, foram projetados 3 experimentos: i) consiste em analisar se o algoritmo codificado consegue convergir para uma solução boa; ii) cópia do primeiro experimento, porém utilizando elitismo, assim é possível ter certeza que o ambiente está se comportando conforme previsto; iii) utilização de diversos padrões para analisar o cenário gerado.

4.1. Experimento I

O primeiro experimento tem como objetivo verificar se o algoritmo consegue convergir, ou seja, se a codificação está convergindo. Sendo assim, ele foi configurado com uma população de 200 indivíduos, sendo que nenhum indivíduo foi copiado para a nova população (elite = 0). O algoritmo de um ponto de cruzamento foi utilizado e a mutação aleatória com taxa de 1%. O critério de parada adotado foi de quantidade de gerações, e neste caso, foi utilizado 1000 gerações. Para avaliar o algoritmo foi utilizado somente 1 padrão (B – que tem somente o chão do cenário). A semente aleatória é 11 - número usado para iniciar o algoritmo gerador de números pseudoaleatórios. A Tabela 2 exibe um resumo das configurações do primeiro experimento.

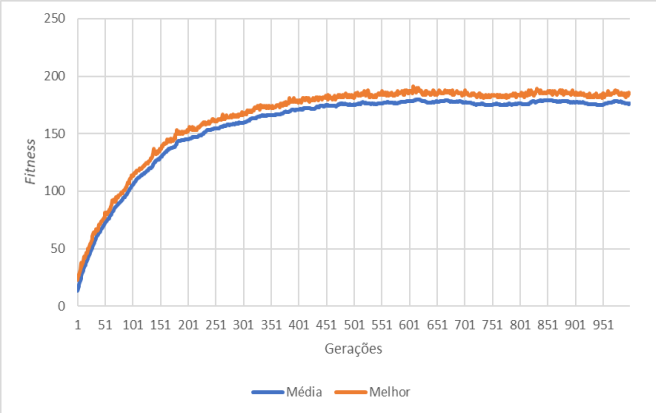
Tabela 2 - Configuração inicial do primeiro experimento.

Variável	Valores
Semente aleatório	11
Tamanho da população	200 indivíduos
Crossover	1 ponto
Taxa de mutação	1%
Elitismo	0%
Critério de parada	1000 gerações
Quantidade de torneio	2
Quantidade de padrões	1

Fonte: Elaborado pelos autores (2020).

O gráfico ilustrado na Figura 6, apresenta a evolução do melhor indivíduo e da média da população. Percebe-se que o algoritmo conseguiu convergir e que a partir da geração 600 ele ficou estagnado por volta de 175 (*fitness*). Contudo, é possível notar uma instabilidade no *fitness* do melhor indivíduo, isso provavelmente acontece por não ter sido usado elitismo.

Figura 6. Gráfico da evolução do *fitness* da população ao longo das gerações no Experimento I.



Fonte: Elaborado pelos autores (2020).

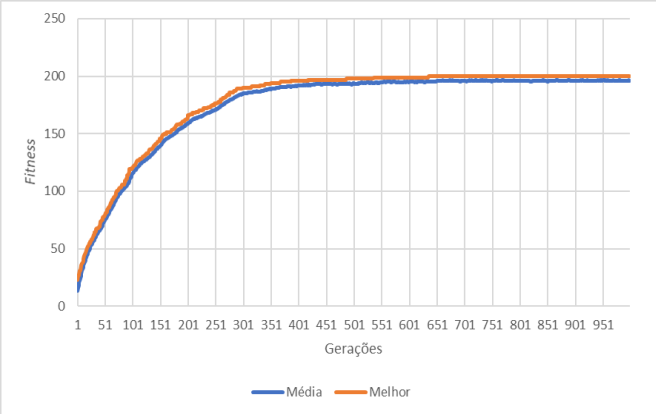
A Figura 7 apresenta uma captura de tela do cenário criado pelo AE no Experimento I. É possível notar, que o cenário chegou próximo do esperado – foi utilizado somente o padrão B, ou seja, era para o cenário ter somente o chão (padrão B).

4.2. Experimento II

O segundo experimento é uma cópia do Experimento I, porém, foi adicionado o elitismo (elite = 10%). Com isso, os 20 melhores indivíduos da geração serão copiados para a próxima geração. Sendo assim, se o algoritmo estiver corretamente codificado, ele irá conseguir convergir mais rápido e talvez chegar na solução ótima.

O gráfico ilustrado na Figura 8, apresenta a evolução do melhor indivíduo e da média da população. Percebe-se que o algoritmo conseguiu convergir mais rápido – por volta da geração 400 - e atingiu a nota máxima de 200. Além disso, é notável a estabilidade e velocidade ao adicionar o operador de elitismos.

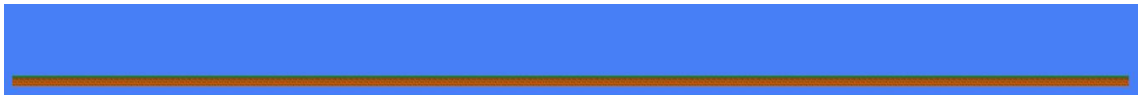
Figura 8. Gráfico da evolução do *fitness* da população ao longo das gerações no Experimento II.



Fonte: Elaborado pelos autores (2020).

A Figura 9 apresenta uma captura de tela do cenário criado pelo AE no Experimento II. É possível notar, que o cenário chegou ao ponto ótimo – foi utilizado somente o padrão B, ou seja, era para o cenário ter somente o chão.

Figura 9. Captura de tela do cenário criado pelo Experimento II.



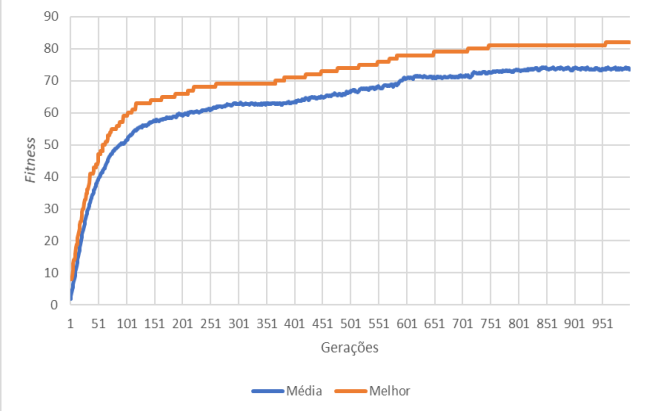
Fonte: Elaborado pelos autores (2020).

4.3. Experimento III

O terceiro experimento é uma evolução do Experimento II, no qual, foi adicionado 8 padrões extraídos da primeira fase do SMB (Tabela 1) e a taxa de mutação foi aumentada para 2%. A taxa de mutação, influencia na variabilidade das soluções adicionando elementos aleatórios no cromossomo. Além disso, foi aumentado a quantidade de indivíduos na população para 500.

O gráfico ilustrado na Figura 10, apresenta a evolução do melhor indivíduo e da média da população para o experimento III. Neste gráfico é possível notar o mesmo comportamento do experimento 2 – ele conseguiu convergir rapidamente e se manteve estável por causa do elitismo. A nota máxima atingida foi de 82 utilizando os padrões da Tabela 1. Vale ressaltar que, muitos outros padrões poderiam ser utilizados, tais como: padrões com inimigos, com tubos, padrões impossíveis de jogar (pontos negativos).

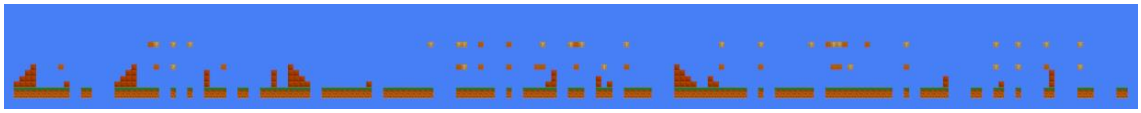
Figura 10. Gráfico da evolução do *fitness* da população ao longo das gerações no Experimento III.



Fonte: Elaborado pelos autores (2020).

A Figura 11 apresenta uma captura de tela do cenário criado pelo AE no Experimento III. Como era esperado pela convergência do AE, o resultado do cenário gerado foi bem interessante e com certeza é possível colocar ele em um jogo de verdade. Vale ressaltar que, o início da fase seria sempre o mesmo e o final também seria sempre o castelo, por isso, essas 2 partes não fazem parte do cromossomo.

Figura 11. Captura de tela do cenário criado pelo Experimento II.



Fonte: Elaborado pelos autores (2020).

5. Conclusão

Este projeto teve como proposta, desenvolver um AE capaz de gerar cenários para um jogo de plataforma (Super Mario Bros). Para isso, foi necessário criar um ambiente do jogo na Unity 3D com a linguagem C#.

O projeto seguiu com o desenvolvimento do cromossomo do AE que pudesse representar uma fase do SMB. Logo em seguida, definiram-se os operadores genéticos e a função de avaliação.

Para concluir, utilizou-se o AE para criar cenários do SMB, definindo de forma empírica configurações dos atributos. Com isso, foi possível testar o AE, obtendo de cenários de forma satisfatória, mostrando que o AE é capaz de gera cenários de forma procedural.

Muitas melhoras podem ser realizadas nestes projetos, assim, pode ser citado: i) codificação e testes de novos operadores de crossover, mutação e seleção; ii) criação e testes de outras funções de avaliação, levando em consideração a inclinação ou espaçamento dos padrões; iii) definição de uma bateria de testes utilizando uma combinação dos parâmetros e operadores; iv) criação de um agente autônomo para jogar e validar a dificuldade do cenário.

Referências Bibliográficas

COSTA JUNIOR, Alberto Rodrigues; LISBOA RAMALHO, Geber. **FERRAMENTA PARA GERAÇÃO PROCEDIMENTAL DE NÍVEIS EM JOGOS INFINITOS**. 2014. Trabalho de conclusão de curso (Graduação em Ciência da Computação) - Universidade Federal de Pernambuco, [S. l.], 2014.

DAHLSKOG, S., TOGELIUS, J.: **Patterns as Objectives for Level Generation.** In: Proceedings of the Second Workshop on Design Patterns in Games, 2013.

DE JONG, Kenneth. **Evolutionary computation: A unified approach**. 2012. GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion [...]. [*S. l.*: *s. n.*], 2012. p. 737–750.

EIBEN, A E; SMITH, James E. **Introduction to Evolutionary Computing**. 2nd ed. [*S. l.*]: Springer Publishing Company, Incorporated, 2015.

FERREIRA, Lucas Nascimento. **procedural de níveis em jogos de quebra-cabeça baseados em física**. 2015. Trabalho de conclusão de curso (Mestrado em Ciência da Computação) - USP, [S. l.], 2015.

HOLLAND, John H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence**. Ann Arbor, MI: University of Michigan Press, 1975.

KIM, Hyun-Tae; KIM, Kyung-Joong. **Hybrid of Rule-based Systems Using Genetic Algorithm to Improve Platform Game**. Procedia Computer Science, ScienceDirect, p. 114-120, 1 dez. 2020.

MARIÑO, Julian; PEREIRA, Leonardo; TOLEDO, Claudio. **Geração Procedural de Conteúdos para Jogos Digitais: Conceitos e Trabalhos Relacionados**. Trilha Principal, [S. l.], p. 1-15, 2 dez. 2020.

VIANA, Breno Maurício de Freitas. **Geração Automática de Níveis de Masmorras com Barreiras para Jogos Digitais**. [S. l.: s. n.], 2019.