# Mecânicas da codificação como elemento da jogabilidade: um estudo inicial sobre ludicidade em narrativas de programação de computadores

**Guilherme Rafael Soares**[1,2]

[1]Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier, 524, Maracanã, Rio de Janeiro, Brasil

[2]Universidade Federal do Recôncavo da Bahia
Rua Rui Barbosa, 710 - Centro - Cruz das
Almas, Bahia, Brasil

guilherme.soares@ufrb.edu.br

***Abstract.*** *Problematization of some digital games that make the use of programming as a diegetic element in the mechanics and aesthetics of their gameplay. Comparing these examples and the application of concepts, we seek here a reflection on playfulness in the practice of computer programming as an element of gameplay, as poetics and as the basis of a research routine that brings together arts, new media and digital literacy.*

***Keywords—*** *creative coding, media literacy, hacker pedagogy, gameplay*

***Resumo.*** *Contextualizamos as problematizações de alguns jogos digitais que propõem o uso da programação como elemento diegético na mecânica e estética de suas jogabilidades. Comparando estes exemplos e a aplicação de conceitos, buscamos aqui uma reflexão sobre ludicidade na prática de programação de computadores como elemento da jogabilidade, como poética e como base de uma rotina de pesquisa que aproxima artes, novas mídias e letramento digital.*

***Palavras-chave—*** *codificação criativa, letramento midiático, pedagogia hacker, jogabilidade*

## 1. Letramento procedural e aspectos narrativos do “brincar de codificar”

Um conceito importante para o que queremos aqui elaborar, já gerou muito debate durante a constituição da área dos game studies: é o que Ian Bogost (2007) chama de retórica procedural ou proceduralidade. O conceito tem bastante relação com aquilo que Jesper Juul (2019) também trata por “regras de jogos de emergência” e é fundamental para entender o argumento de que os videogames fornecem experiências de imersão governadas por regras originais que fazem surgir contingências e padrões em uma ficção interativa.

Apesar deste argumento ser por vezes atacado por ser usado de maneira desajeitada para defender pontos de vistas que exageradamente querem dizer que os sistemas de regras e competição precedem os elementos narrativos dos jogos [FERREIRA 2017], é importante reconhecer que há uma intenção relevante neste esforço de Bogost: pensar os jogos como aliados de um letramento lúdico na codificação e a proceduralidade. A retórica procedural poderia ser vista assim como uma espécie de rotina criativa que demanda tanto do game designer em seu projeto como do próprio jogador em sua experiência imersiva alguma vontade ou expectativa de pensar como um programador [BOGOST 2009].

James Paul Gee produz sínteses essenciais para pensarmos os games no contexto do letramento. Algumas categorizações que definiu podem nos ajudar a entender algumas vantagens no aprendizado ou primeiro contato com a programação nos games, como gostaríamos de definir a partir daqui. Queremos destacar neste momento o que ele chama de "Performance anterior a competência"[GEE 2009] : a possibilidade de aprender uma habilidade na prática, intuitivamente – por exemplo a capacidade de aprender com as mecânicas dos jogos as suas regras e lógica de operação - para em seguida desta experiência conseguir acompanhar qualquer tipo de instrução escrita e detalhada sobre esta experiência com muito mais profundidade.

Pensado a possibilidade de um letramento na codificação que leve em conta esses aspectos de proceduralidade da mídia game e considerando este tipo de aprendizado uma possibilidade de "performance anterior à competência" – queremos aqui exemplificar experiências separando em dois paradigmas:

*a) Brincar aprendendo:* O jogo oferece a possibilidade de programar como parte da narrativa e fornece algum tipo de acesso a este treinamento durante o jogar. É um caso bem exemplar de "performance anterior a competência" [GEE 2009] - o aprendizado estimulado pelos objetivos do jogo.

*b) Aprender para brincar:* Codificar é opcional e parte da usabilidade do jogo. Um passo adiante dentro da própria programação do jogo e de suas modificações. Neste caso a programação não precisaria ser necessariamente parte da trama. Mas no exemplo que usamos aqui, isso também pode acontecer.

## 2. Brincar aprendendo

### 2.1. Narrativa do hacker mítico: "else heart.break( )" e "Hacknet"

Um primeiro padrão facilmente reconhecível são jogos onde há a performatividade clichê de um hacker no sentido romântico, messiânico e fora-da-lei. Em missões de espionagem ou quebra de regras do sistema, ele transforma o ambiente com suas interações subversivas.

A discussão sobre a relatividade deste lugar-comum, e a problematização da ética hacker como definidora de uma classe trabalhadora real e atuante [WARK 2021] que desenvolveu uma rotina de curiosidade e colaboração poderia render uma digressão, mas fica para um foco maior em outros artigos. Aqui vamos dar uma atenção maior às mecânicas destes jogos e como eles proporcionam uma experiência de contato com conceitos de programação de computadores dentro da jogabilidade destes.

O jogo Hacknet (2015) fornece um exemplo de jogo focado na ação em um terminal de comandos, simulando a experiência de usuário de ambiente Unix (similar aos terminais de comando de Linux e MacOSX) com algumas representações gráficas e também clicáveis com mouse.

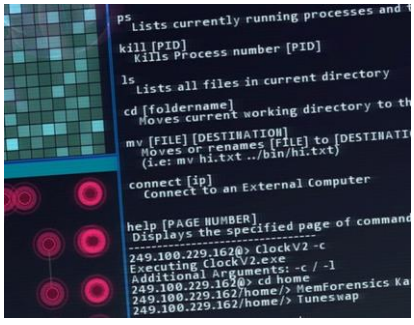


**Figura 1. O jogo Hacknet e os comandos nativos da sintaxe Unix**

Alguns comandos são similares aos usados para navegação de diretórios, processos, árvores de arquivos e pastas, diferenciação de formato texto e formato binário, simulação de ambiente de interação entre diferentes computadores em uma rede TCP/IP - protocolo usado até hoje como majoritário em redes de roteadores domésticos e na Internet. Comandos como ls, cd , ps, scp (Figura 1) - utilizados cotidianamente nas rotinas de administração de sistemas e servidores baseados no padrão UNIX estão ali, mesmo que alguns deles figurem fazendo operações “mágicas” que demandariam muito mais variáveis.

Por outro lado, a experiência desenvolve uma familiaridade de fato bastante intuitiva para quem já lidou com sistemas de servidores web ou similares. Ao mesmo tempo proporciona uma mecânica de curiosidade e acesso a tais alegorias para quem nunca teve a experiência.

Apesar da navegação de arquivos e IPs mostrar uma dinâmica bastante realista, este jogo não apresenta uma mecânica que estimule a programação propriamente dita, e sim a navegação de arquivos, máquinas e execução de comandos de um repertório entre instalados no ambiente original ou arquivos encontrados no decorrer do jogo.

Em else.Heart.Break()(2015) temos um caso um tanto diferente e bastante singular. Neste jogo temos um foco grande na interação entre um ambiente 3D bastante colorido e estilizado, populado por personagens com quem você pode interagir ao estilo dos clássicos adventures point-and-click . O personagem está de aniversário e sai em uma caminhada por uma cidade portuária em busca de atender a uma chamada para um emprego precário. No entanto, à medida que ele vai conversando com pessoas, explorando o ambiente entre cafés, bares, casas de vizinhos, metrô, usinas, ele encontra a trama de um grupo de hackers e seus dispositivos espalhados pelo mapa - uma série de disquetes com pistas de como manipular uma linguagem de programação desenvolvida especialmente para o jogo. Além das ações que estão dentro da narrativa  e objetivo deste - que é ajudar o grupo de hackers da cidade a derrubar um ministério corrupto. Há neste jogo uma grande possibilidade de experimento livre de deriva do tipo mundo aberto.

**Figura 2. A rotina de programação embutida no estilo de vida de else.Heart.Break()**

É possível experimentar de diversas maneiras com a linguagem - hackear o cenário, o tempo do jogo - mas para além disso é possível também experimentar o uso da linguagens para criar programas dentro dos computadores do jogo, a ponto de haver na comunidade uma cena de minigames desenvolvidos para o arcade do boteco do jogo e a possibilidade de interferir em camadas de jogabilidade da história.

Ações divertidas como hackear um cigarro para o tempo passar mais rápido, ou viajar pelo metaverso dentro da rede de computadores do jogo estimulam um entendimento dos padrões de iteração em listas, funções e parâmetros do jogo, trazendo uma experiência de programação de fato (Figura 2) jogabilidade ou atualizações no cenário, personagens e diálogos.

## 2.2. Narrativa do trabalho como desenvolvedor de sistemas

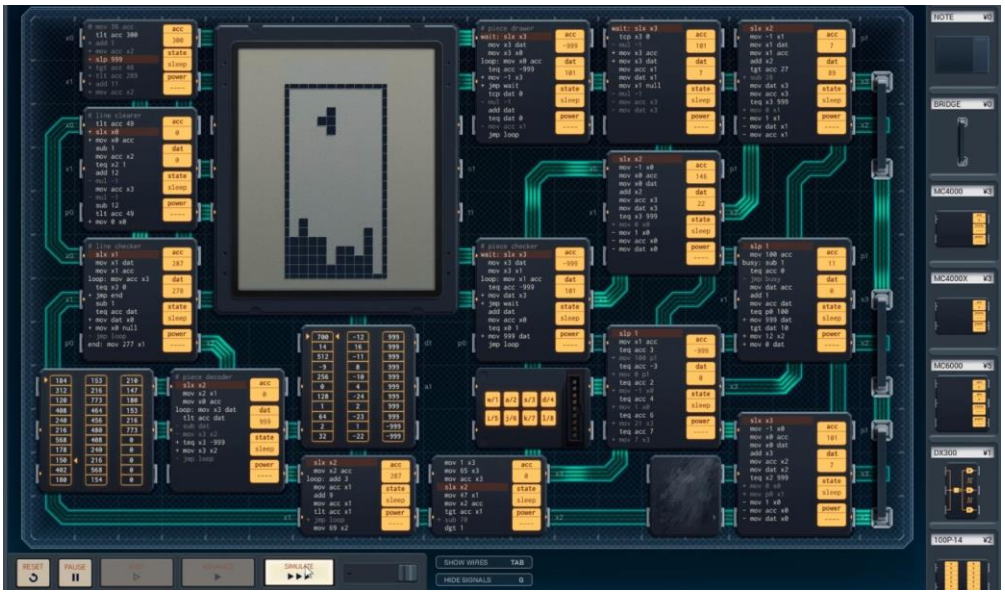

**Figura 3. A rotina de programação de hardware em Shenzen I/O**

Um segundo padrão reconhecível são os jogos de programar onde o protagonista trabalha em missões para firmas de software ou para a indústria de automação. Shenzen I/O faz parte de uma série de jogos interessantes e autorais do desenvolvedor Zach Barth e seu estúdio Zachthronics. Neste jogo o protagonista trabalha para uma fábrica chinesa de hardware e fica recebendo progressivamente novos projetos de hardware encomendados para você por email como freelancer da indústria chinesa. A jogabilidade é baseada em conectar módulos de hardware (Figura 3) e sincronizar sinais bit a bit, reconhecendo padrões de oscilação e utilizando uma linguagem assembler bastante realista. É interessante pois cada vez mais os programadores estão trabalhando em camadas de abstração mais altas como sistemas de servidores ou clientes web ou aplicativos de sistemas operacionais de computadores e mobiles. O trabalho com assembler é algo realmente aplicado para quem trabalha na camada mais baixa da indústria de hardware ou em projetos de microprocessadores. A linguagem do jogo é bastante completa e trabalha com estruturas de loops, saltos, operações básicas e alocação de memória, processamentos paralelos e saída em tela ou sonora.

Esta linguagem é completa o suficiente para gerar uma cena interessante de provas de conceito de aplicações mais complexas, como fazer rodar uma versão de Tetris ou sintetizadores FM dentro do jogo.

Em while True: learn()(2015) há uma estética mais colorida e cartunesca, que pode ter uma apelo maior ao público infantil, apesar de tratar de um tema bastante específico da computação e que pode gerar curiosidade geral - o aprendizado de máquina. A história é bastante narrativa e baseada na ideia de que o gato do programador conseguiu ajudar a organizar algoritmos de aprendizado de máquina do seu dono. Então ambos se envolvem numa aventura freelancer pelo mercado da Inteligência Artificial. Este jogo utiliza estruturas de grafos em programação visual de fluxo de dados e tem uma jogabilidade baseada em encomendas similar ao jogo descrito anteriormente.

Apesar de mergulhar nos algoritmos e rotinas, em while True: Learn não temos uma linguagem de programação escrita mais próxima das conhecidas, apesar de que uma inserção progressiva de novos módulos e métodos de otimização dos algoritmos introduz questões de treinamento e aprendizado de máquina que podem gerar interesse no tema. O jogo tem também uma mecânica mais focada na ideia do trabalho do programador como empreendedor - fazendo-o investir em compra e venda de ações dos produtos que desenvolve e investindo dinheiro em módulos de equipamento para acelerar e otimizar seus testes.

Em Human Resource Machine (2015) há uma situação com algumas características das duas anteriores. A jogabilidade é também mais cartunesca, com uma história baseada na organização de algoritmos e busca em uma esteira. Com uma linguagem de programação gráfica que vai introduzindo módulos a cada novo desafio, Human Resource Machine tem uma linguagem simples e similar a uma rotina de assembler (Figura 4) - fornece uma noção de gerenciamento de espaço de memória temporária e um depurador passo-a-passo para otimização e conserto dos algoritmos durante o teste

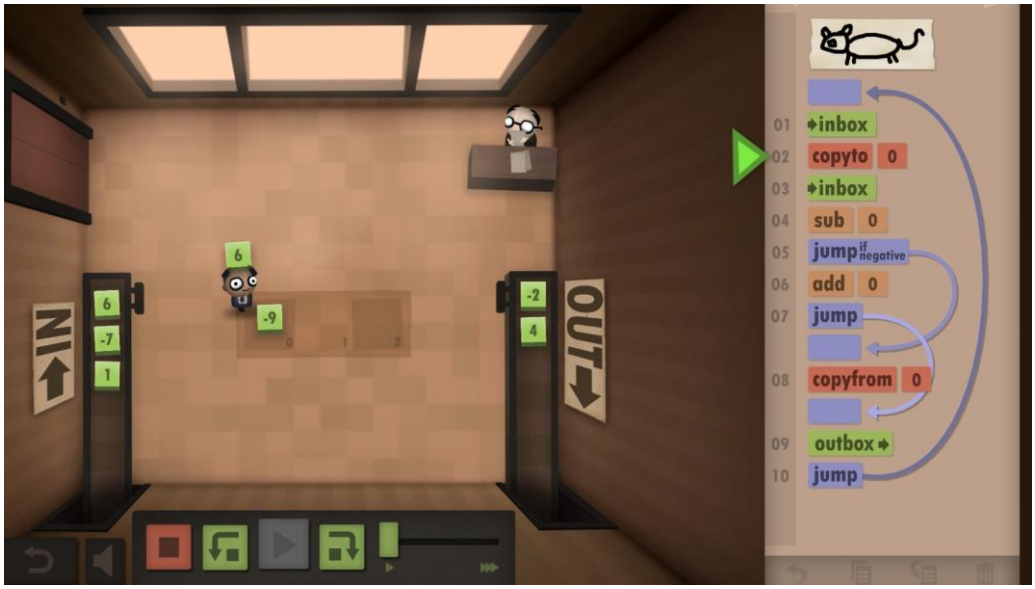


**Figura 4. Proximidade da linguagem assembler em Human Resource Machine**

## 3. Aprender para Brincar

### 3.1. Puzzles e a metalinguagem sobre codificar

Voltando à classificação de Jesper Juul (2019) sobre diferentes matizes de inserção da narrativa versus abstração da jogabilidade nas regras e objetivos, temos aqui um ponto fora da curva interessante - o jogo Baba is You(2020). Este é um puzzle similar ao clássico game japonês dos anos 80 Sokoban(1982). Em Sokoban você basicamente empurra e posiciona caixas para liberar espaço, encontrar itens e organizar pilhas em um estoque. É um puzzle que de certa forma originalmente não é dependente de sua narrativa. Sokoban (倉庫番 ) é o nome de um operário de armazém que organiza estoques - porém isso não é relevante para que as caixas estejam no lugar certo. O que é mais importante em Sokoban é sua estratégia para abrir caminhos e encaixar as peças e seu sistema de contingências e estratégias de adaptação emerge naturalmente das regras iniciais.

Baba is you conseguiu o grande feito de pegar essa ideia e inserir no jogo uma lógica abstrata de jogos de sintaxe criando algoritmos de solução do puzzle. Você move blocos de predicados e vai redefinindo durante o jogo o papel das peças. Por exemplo: você pode iniciar com as regras: Baba is you (Baba é você), Wall is Stop (Muro é Parada) , Flag is Win (Bandeira é Vitória). Se você mover o bloco Wall que está compondo o predicado “Wall is Stop” (Figura 5), o muro passa a não ser mais algo que paralisa o personagem, e ele consegue atravessar a parede. Essas soluções vão ficando cada vez mais complexas e nem sempre há apenas uma solução possível.

Uma outra grande possibilidade que comprova a proximidade entre a experiência de Baba is You e a programação de computadores é a distribuição de um editor de fases que aceita contribuições e modificações de terceiros, disponíveis para download gratuito na interface do jogo. Isso criou uma cena bastante interessante de pessoas usando Baba is You como linguagem para criar outros jogos ou provas de conceito. Assim como em Shenzen I/O é possível encontrar diversos experimentos como Pong (Figura 6), Tetris, PacMan ou coisas como um contador binário, sequenciadores de samplers musicais ou uma máquina de turing. Apesar de isso obviamente não ser o suficiente para transformar os blocos de Baba Is You em algum novo paradigma de programação, isso demonstra o potencial deste tipo de jogo para incentivo do pensamento lógico abstrato essencial para a codificação.


**Figura 5. Quebra-cabeças lógicos de sintaxe em Baba is You**

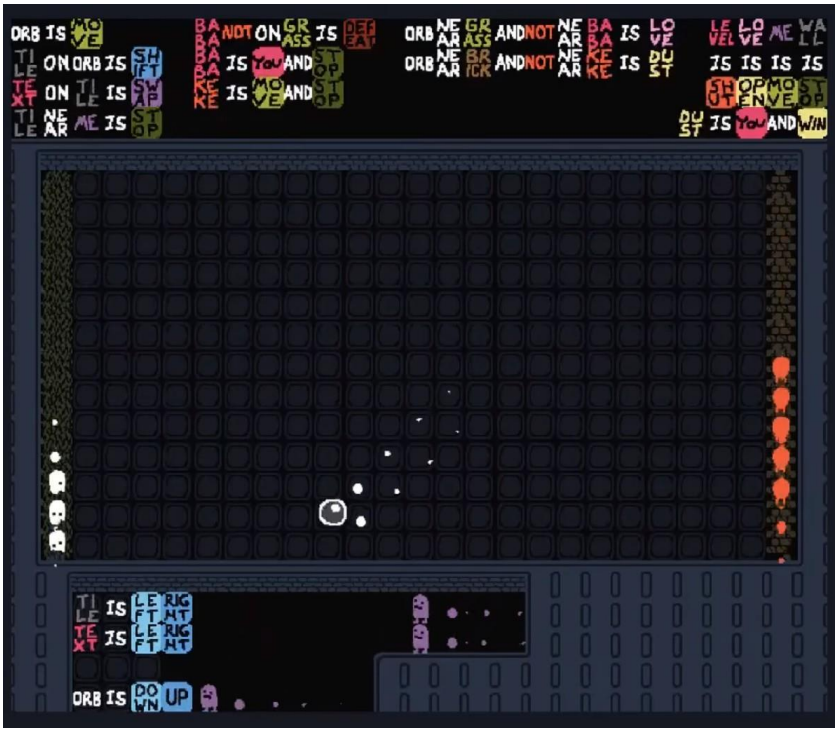


**Figura 6. Rotina do jogo Pong feita com o jogo Baba is You como linguagem**

## 3.2. Codificação Criativa como hacking da jogabilidade

Voltando à classificação de Jesper Juul (2019) sobre diferentes matizes de inserção da narrativa versus abstração da jogabilidade nas regras e objetivos temos aqui um ponto importante.

Um caminho intermediário para pensar o passo além do jogo, onde a narrativa inclui a manipulação de linguagens de programação ou blocos de algoritmos, é pensar casos onde a programação é parte opcional na usabilidade deste, apesar de seu uso ter o potencial de influir diretamente na jogabilidade. Qualquer sistema de cheats - códigos para libertar certas funções bloqueadas na jogabilidade normal do jogo - poderia ser pensado como um primeiro passo nesta direção. Mas interessante seria mesmo encontrar exemplos onde é possível criar rotinas mais complexas e manipular entradas e saídas de dados durante do tempo do jogo. Um exemplo bastante popular é o jogo Minecraft e suas variações, como a versão open source Minetest.

As modificações mais comuns são rotinas para intervenção direta na modelagem dos mundos - com repetições, cópias, empilhamentos, redimensionamento estendidas pelo mod World Edit . Alguns outros mods como ComputerCraft podem inserir dentro do jogo computadores funcionais que permitem a criação de narrativas que incluem usar scripts lua para dar saídas de texto, interferir nos blocos do jogo. É possível criar portas com senha ou outras automações que quando usadas em conjunto com os blocos de portas lógicas e alavancas do jogo (as Redstone no Minecraft e o mod Mesecons no Minest) podem servir diretamente para o design do mundo incluso na jogabilidade.

Em casos mais complexos é possível encontrar também mods que são hackings nas entradas e saídas do jogo, como o uso de uma camera Kinect para atualizar esculturas feitas com os blocos do mundo generativo (Figura 7) ou a possibilidade de usar um telefone móvel para mandar mensagens de vídeo dentro de um celular esculpido jogo.

Este tipo de modding (modificações de usuário), no entanto, apesar de extremamente estimulante para os jogadores interessados em uma cena de compartilhamentos de modificações, demanda um envolvimento com codificação que já inclui a experiência em manipulação do sistema - independente da simples manipulação de blocos e textos dentro do próprio jogo.

O acesso a códigos e aos contextos de sua interpretação geralmente demandam uma curva de aprendizado acentuada no início. Afinal isso demanda busca por estes ambientes, sistemas, serviços de provedores. Este estímulo e possibilidade de acesso imediato da codificação dentro do jogo (e a partir dele, com tutoriais e ferramentas de modificação) podem ser este primeiro passo. Anete Vee (2019) , em seu livro sobre letramento na codificação, relembra o fato de que esta dificuldade de acesso a um ambiente inicial que valorize as interfaces de comando textual e para acesso direto ao sistema (como os terminais Unix e o prompt DOS) por um lado adaptaram o usuário a interfaces de poucos passos, mas por outro afastaram-no de uma rotina mais próxima da interação direta com a operação dos sistemas. A valorização estética e narrativa destes elementos favorece este tipo de reflexão.

Esse caminho nos faz refletir também sobre a possibilidade de pensar os limites deste “brincar com códigos” para além do escopo dos jogos digitais ou dos seus hackings. E sim como uma prática cotidiana de aprendizado e adaptação. No esforço pela construção de uma “epistemologia da gambiarra”, os pesquisadores José Messias e Ivan Mussa (2020) em certa altura problematizam a questão das diferentes camadas de complexidade que o hardware atual apresenta, tendo que lidar com diferentes protocolos, em diversas instâncias paralelas de funcionamento do sistema operacional e seus softwares, fazendo com que estes “glitches que naturalmente emergem do sistema” possam também figurar como uma espécie de brecha para a invenção.

**Figura 7. Hacking com a câmera Kinect no Minecraft – ambiente estimula a codificação criativa**

## 4. Considerações finais

Almejamos, com o início da presente sistematização e análise deste repertório de jogos uma abertura para um mapeamento de rotinas lúdicas de codificação criativa inspiradas neste uso da programação como parte da jogabilidade. O presente artigo ainda foca ainda mais na descrição de algumas mecânicas do que um aprofundamento sobre a possibilidade de letramento na codificação estimulada por estes jogos – especulação que fica para sua continuidade.

A pesquisa iniciada com este artigo buscou uma classificação inicial para estes repertórios partindo uma aproximação da bibliografia dos estudos culturais do software (software studies) e dos estudos dos games como mídia e plataforma (game studies e platform studies). Pretendemos pensar partindo daqui em uma abertura para convergência de dois problemas importantes sobre os games no letramento digital e midiático: a perspectiva de consideração de um letramento na codificação e o papel da programação, tanto na narrativa quanto na usabilidade dos jogos, como uma possibilidade de atividade lúdica para este tipo de letramento. Por um lado, teóricos das mídias como Manovich (2013) e Bogost (2007) alertam para a urgência do problema de interpretação da materialidade das mídias (mediado também por um entendimento mais aprofundado dos códigos). Por outro, faz-se urgente a consideração de uma atenção mais implicada e sensível aos contextos culturais que fizeram da mídia videogame a ponte entre conteúdos audiovisuais e as origens das interfaces gráficas computacionais.

É possível perceber já na análise aqui presente que muitas destas etapas podem ser reveladas já na simples experiência estética e lúdica de contato com a codificação como parte da narrativa de jogos. Em próximas etapas desta pesquisa, pretendemos refinar e organizar ainda mais este repertório tanto para uma referência no letramento para a codificação quanto na possibilidade de fomentar uma codificação de novos jogos autorais que levem em conta esta proposta e realimentem o ciclo.

## Referências


BERS, M. (2020) Coding as a playground: Programming and computational thinking in the early childhood classroom. Routledge.

BOGOST, I. (2009) Procedural Literacy. Persuasive Games: The Expressive Power of Videogames, p. 233-260, 2007. GEE, James Paul. Bons video games e boa aprendizagem. Perspectiva, v. 27, n. 1, p. 167-178.

FERREIRA, E. (2017) Retórica processual e experiência videolúdica nos indie games. Falcão, T., & Marques, D.(Orgs.), p. 43-71.

GEE, J. (2009) Bons video games e boa aprendizagem. Perspectiva, v. 27, n. 1, p. 167-178.

MONTFORT, N. (2014) 10 Print Chr $(205. 5+ rnd (1));: Goto 10. MIT Press.

REGIS, F. (2020) Letramentos e mídias: sintonizando com corpo, tecnologia e afetos. Revista Contracampo, v. 39, n. 2.

MANOVICH, L. (2013) Software takes command. A&C Black.

MATEAS, M. (2005) Procedural literacy: educating the new media practitioner. On the Horizon.

MESSIAS, J. e MUSSA, I. (2020) Por uma epistemologia da gambiarra:: invenção, complexidade e paradoxo nos objetos técnicos digitais. MATRIZes, v. 14, n. 1, p. 173-192, 2020.

KITTLER, F. (2017) A verdade do Mundo Técnico: ensaios sobre a genealogia da atualidade. Rio de Janeiro: Contraponto.

WARK, M. (2021) Capital Is Dead: is this something worse?. Verso books.

VEE, A. (2019) Coding literacy: How computer programming is changing writing. Mit Press.