Roguelikes: conexão jogador-máquina nos ambientes digitais de videogames [1]

Ivan Mussa[2]
Universidade do Estado do Rio de Janeiro (UERJ)

Resumo

Este trabalho apresenta um problema de pesquisa referente ao modo como compreendemos os videogames e seus usos. Percorrendo a extensa gama de experimentações possíveis com jogos eletrônicos, pode-se perceber algumas que anseiam construir ambientes amplos e dinâmicos. Nestes jogos, o jogador conecta-se a um espaço habitado por entes autônomos e itens manipuláveis, que fazem emergir eventos imprevisíveis. Neste contexto, nota-se a ausência de preocupação com a dimensão comunicacional desta conexão. Sendo assim, o objetivo deste texto é explorar um gênero de videogames – os *roguelikes* – com o intuito de investigar o modo como produzem ambientes. Este objetivo destrincha-se em três etapas: uma arqueologia dos jogos que compõem o gênero; um olhar microscópico sobre o funcionamento interno dos seus ambientes; e, finalmente, a busca por um pensamento que dê conta deste tipo de comunicação.

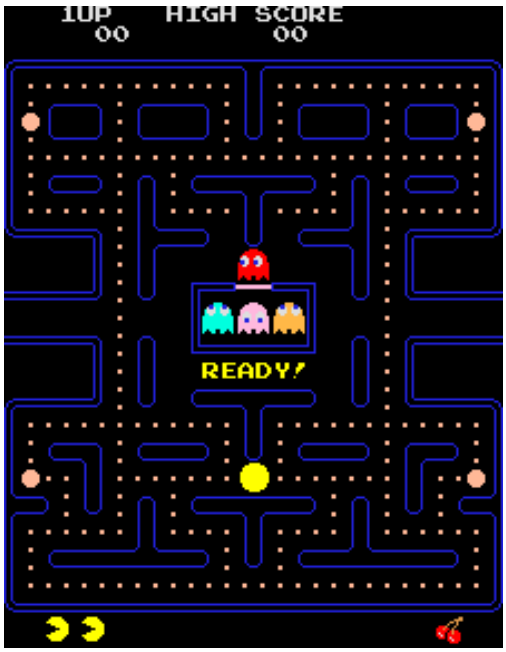
Introdução

É possível considerar o jogo *Pac-Man* um ambiente? A máquina *arcade* que em 1980, pela primeira vez, proporcionou esta experiência de jogo encaixaria-se nesta categoria, da mesma forma que uma cidade populosa ou um ecossistema aquático? A primeira e mais evidente característica que separa o jogo dos outros exemplos escolhidos é sua dimensão digital. Protegidos pela cabine e conectados por sistemas de hardware, os circuitos elétricos do computador que rodavam a primeira versão de *Pac-Man* davam origem a dados binários. Sequências de 0 e 1 representavam números, que por suas vezes designavam cores para os pixels da tela e frequências de onda para os canais de som.

---

[1] Trabalho apresentado no Grupo de Trabalho 6 – COMUNICAÇÃO, CONSUMO E SUBJETIVIDADE, do 5º Encontro de GTs - Comunicon, realizado nos dias 5, 6 e 7 de outubro de 2015.
[2] Doutorando em Comunicação pela Universidade do Estado do Rio de Janeiro (UERJ) E-mail: ivanmussa@gmail.com.

Além destas funções, a caixa-preta binária de *Pac-Man* codificava parâmetros para input e output de informações. Quando o *joystick* era movido para a direita, o programa reorganizava os dados armazenados no sistema, simulando o movimento do personagem. Este truque de manipulação direta articulado pela interface adicionava um nível importante à experiência: a capacidade de "locomoção" no "espaço". Na realidade, essa locomoção não passava de pixels trocando de cor segundo regras organizadas para causar a ilusão de deslocamento. E esta espacialidade originava-se do entrelaçamento dos pixels da tela com os dados armazenados no disco rígido: um truque de programação. Ainda assim, os jogadores percebiam imediatamente, ao jogar, que estavam controlando um objeto que se movia ao longo de um labirinto na tela.



Pac-Man (1980)

As maquinações internas dos códigos de programação funcionavam através de regras obscuras demais para serem interpretadas por leigos. Mas um conjunto de regras mais evidente era mais facilmente identificável através da interação: movimentar o personagem sobre os pontos brancos menores fazia-os desaparecer. Se um dos quatro fantasmas coloridos que também se moviam pelo labirinto encostasse no personagem controlado pelo jogador, o jogo era interrompido. Quando os pontos brancos maiores

eram alcançados, os fantasmas mudavam seu padrão de movimento e suas cores, e podiam ser eliminados.

Estas regras mais compreensíveis, em conjunto com o design visual e sonoro do jogo, constituem uma simulação. Esta simulação faz referências a processos que, de alguma forma, são assimiláveis. Mesmo que esta não seja uma situação que observemos cotidianamente, sabemos que o ser redondo e amarelo que controlamos acabou de devorar um fantasma atordoado. Nosso personagem – ou *avatar* – tem uma habilidade, um potencial de alterar os dados a sua volta.

Os outros seres moventes – os fantasmas – também são controlados por regras que estipulam formas de afetar aquilo que os rodeia. Sua função é dificultar a missão dada ao jogador: esgotar os pontos brancos espalhados pelo labirinto. As capacidades concedidas através da programação a cada fantasma desenvolvem uma espécie de comportamento particular para cada um deles. O fantasma vermelho sempre busca o caminho mais curto entre ele e o ponto em que jogador está. O fantasma rosa tenta prever para onde o jogador está indo, fazendo um caminho mais longo, mas que tenta encurralá-lo.

Desta forma, temos: Primeiro, um espaço em forma de labirinto no qual agentes podem se locomover; segundo, agentes estes que possuem comportamentos individuais e autônomos; terceiro, jogador e personagens com capacidades de alterar o estado dos demais componentes do sistema. *Pac-Man*, descrito desta forma, começa a se parecer cada vez mais com um ambiente. Obviamente, se compararmos novamente a uma cidade, por exemplo, o manancial de interações possíveis é demasiado raso. Mas mesmo neste jogo relativamente simples parece existir um estágio incipiente de ambiente digital capaz de produzir experiências lúdicas.

Este artigo pretende apresentar um problema de pesquisa que nasce deste modo de olhar os videogames. Ao longo de sua história, os jogos eletrônicos lançaram tentativas recorrentes de povoar um espaço virtual com agentes autônomos, objetos acionáveis, estruturas espaciais e todos os eventos que podem surgir da colisão entre estes componentes. Propomos que a intenção subjacente nesta prática é dar origem a

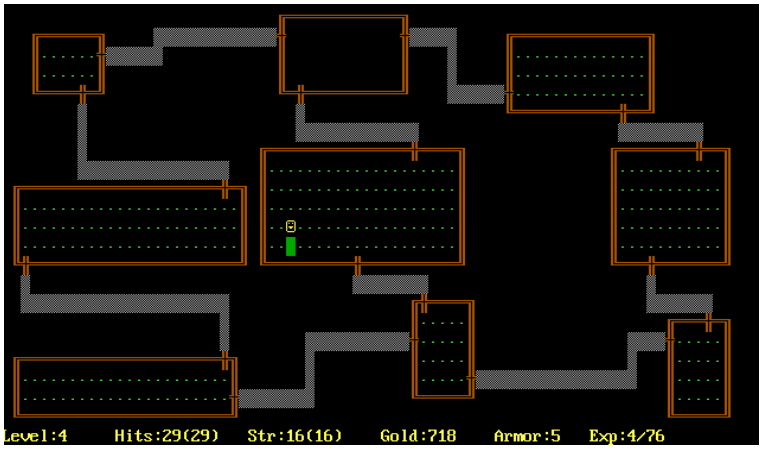
ambientes. Há muitas formas de realizar esta intenção, e uma delas é modelando cuidadosamente um espaço navegável – como o labirinto de *Pac-Man*. Mas é possível utilizar as propriedades computacionais para fazer um programa que gera o ambiente de forma autônoma.

No mesmo ano de 1980, quando Pac-Man apresentou ao público seu labirinto pré-estruturado, um outro jogo eletrônico se popularizou com uma premissa parecida. *Rogue* também ostentava labirintos povoados por inimigos que ameaçavam o jogador. Uma diferença crucial, no entanto, é o fato de que, cada vez que o jogo era ligado, o labirinto era diferente. Não só o labirinto, mas os inimigos, armas e armadilhas trocavam de posição e de características. Michael Toy, Glen Wichman e Ken Arnold desenvolveram o jogo de forma amadora nos *mainframes* da Universidade da Califórnia. Sua inspiração eram os jogos de RPG da série *Dungeons & Dragons*, nos quais os personagens criados e imaginados pelos jogadores se aprofundavam em cavernas desconhecidas para roubar tesouros de dragões ou derrotar *trolls* que assombravam arredores de vilas. Nestas aventuras, o papel de narrador e, muitas vezes, de criador dos ambientes imaginários era de um jogador especializado, chamado *game master* (ou simplesmente "GM"). *Rogue*, enquanto software, reproduzia o papel do GM, elaborando labirintos, povoando-os com seres vivos e não vivos, desafiando o jogador a aprofundar-se nas cavernas digitais que nunca repetiam sua estrutura.

Toy, Wichman e Arnold, a esta altura, provavelmente não haviam jogado *Pac-Man*. Mas jogaram *Colossal Cave Adventure* (1976), jogo baseado em texto que descrevia uma caverna aberta à exploração. As bifurcações labirínticas do jogo foram cuidadosamente modeladas pelo programador e explorador de cavernas Will Crowther, que se baseou em parte de um sistema subterrâneo real, a Mammoth Cave, localizada no estado americano de Kentucky. Uma vez experimentado, o jogo deixava de esconder mistérios: as ameaças não mais surpreendiam e os trajetos mais eficientes eram sempre os mesmos.

Ao programar *Rogue*, os universitários garantiram que seu jogo não teria a mesma desvantagem: cada vez que o jogador rodasse o programa, a caverna seria

diferente. A dupla não atingiu esta meta criando três ou quatro variações. Em tese, a quantidade de configurações de cavernas possíveis em *Rogue* é tão numerosa, que os milhares de jogadores que as exploraram (e continuam explorando) viram apenas uma fração do que é possível com seu software.



Rogue (1980)

Rogue, portanto, possui uma camada de simulação a mais que *Pac-Man*: ele não é um espaço fixo desenhado por pixels na tela. É, na verdade, uma máquina de gerar ambientes. Através de um conjunto de regras algorítmicas, produz variações infindáveis a partir de uma série de diretrizes. Esta forma de criar ambientes lúdicos consolidou-se gradualmente ao longo dos anos, sendo incrementada gradualmente. *Rogue*, nesse contexto, inspirou mais do que homenagens ou imitações: deu nome a um gênero, que é chamado pela comunidade de jogadores e desenvolvedores de "*roguelike*".

Ao longo da década de 1980, qualquer jogo que possuísse gráficos com códigos alfanuméricos, espaços gerados de forma automatizada, dificuldade extrema e outras exigências incomuns, era reconhecido como parte do movimento de jogos *roguelike* (tradução literal: "parecidos com *Rogue*"). Em um *roguelike* clássico, não é possível salvar o progresso: se o jogador deixa o personagem morrer, este é um evento permanente, sendo necessário recomeçar do zero. Os labirintos são sempre construídos pelo código do software, nunca arquitetados e desenhados por um humano (o que

garante variabilidade gerada matematicamente). O processo do jogo funciona por turnos, como no xadrez: o jogador executa uma ação e o mundo do jogo move suas "peças".

As coisas seriam simples se estas convenções não tivessem sido esticadas e entortadas por jogos que, aos poucos, foram remodelando o gênero: códigos alfanuméricos transfiguraram-se em modelos 2D e 3D, cavernas viraram cidades, batalhas ritmadas por turnos passaram a desobedecer o ritmo e movimentarem-se de forma contínua, exigindo mais reflexo que planejamento e paciência.

Este problema pode ser abordado através de três etapas. Primeiro, é preciso mergulhar na história do gênero *roguelike*, rastreando suas metamorfoses e as relações que estabeleceu com outros estilos de jogo. Segundo, é necessário aplicar um *zoom* neste fenômeno, lançando olhares sobre as maquinações internas dos ambientes de jogos específicos. Este olhar microscópico tem o objetivo de explicitar os funcionamentos distintos de ambientes do mesmo gênero de jogo. E, finalmente, pode-se reunir as ideias abordadas nas etapas anteriores sob um ponto de vista comunicacional. Que inconsistências e tensões são identificáveis entre a concepção clássica de comunicação e os fenômenos que emergem da conexão entre seres humanos e ambientes digitais lúdicos?

Arqueologia roguelike

Um gênero de videogame, a princípio, pode ser compreendido como uma reunião de regras respeitadas por uma linhagem de jogos. Jogos do gênero "plataforma" como *Super Mario Bros.* (1985) e *Sonic the Hedgehog* (1991) fazem uso de premissas correlatas: personagens devem ser conduzidos até o final de um percurso, evitando obstáculos, armadilhas e inimigos – e eventualmente coletando recompensas. A movimentação é executada através do controle da velocidade do personagem, e quase sempre o jogador precisa fazê-lo saltar para alcançar áreas mais ou menos elevadas (daí o nome do gênero).

Isto não quer dizer que jogos de plataforma não tenham mudado desde sua incipiência. Pelo contrário: se é possível observar a resistência de grande parte de suas premissas, elas atuam em contextos claramente distintos. Adaptaram-se, por exemplo, aos espaços tridimensionais modelados com polígonos, que só fizeram sua estréia no mercado comercial em 1992. A transição pode ser sentida de forma evidente em *Super Mario 64* (1996). Outras variáveis técnicas interferiram nos moldes do gênero, como a capacidade de processar mais objetos em movimento na tela, a simulação mais fiel de grandezas físicas, etc.

É possível propor, ao observar este tipo de metamorfose, que gêneros são pelo menos parcialmente fluidos. São convenções que sofrem influências não só tecnológicas, mas também estéticas, econômicas e culturais. Jogos surgem como projetos e precisam se adaptar neste ecossistema hostil para chegarem ao público. Às vezes saem desta luta irreconhecíveis, muito diferentes dos seus ancestrais do mesmo gênero. Mesmo assim são reflexos, também, de seu código genético "original", por assim dizer. Seria possível traçar a árvore geneálogica do gênero *roguelike*, bem como rastrear as mutações que se destacaram a partir da influência do ecossistema no qual ele se desenvolveu?

Um ponto de partida possível é justamente o jogo *Rogue*, que em uma análise preliminar já revela pelo menos quatro ancestrais. Um de 1979, *Dungeon*, encena elementos do que seu descendente faria um ano depois. Dois deles de 1978: *Dungeon Campaign* (DC) e *Beneath Apple Manor* (*BAP*), cada um com traços próprios e que fazem lembrar *Rogue*, dois anos antes deste existir. E em algum momento entre 1976 e 1978, *Dragon Maze* é lançado para o computador *Apple II*. Um jogo simples, mas que conseguiu chamar a atenção dos criadores de *DC* e *BAP*. Ambos já declararam em entrevistas que foram diretamente influenciados pelo jogo do programador Gary J. Shannon[3].

[3] Fonte: http://crpgaddict.blogspot.com.br/2012/12/game-79-beneath-apple-manor-1978.html.

Parece existir uma dinâmica na qual os jogos que surgem bebem de seus predecessores em alguns aspectos. Estes predecessores, por suas vezes, inspiraram-se em outras fontes. No entanto, não se trata apenas de uma evolução cronológica, mas de uma combinação criativa de recursos disponíveis no contexto de criação de um dado jogo. Em algum momento, diferentes combinações de elementos lúdicos gerarão ambientes com capacidades novas – e assim o processo reinicia seu ciclo. Por exemplo, quando o gênero *roguelike* cruza em sua teia genealógica com o gênero plataforma, resulta em jogos como *Spelunky* (2009). Este por sua vez, abre uma nova brecha para mais jogos plataforma-*roguelike*, criando um tipo de experiência lúdica com potencialidades próprias.

Visão microscópica: O funcionamento dos ambientes

Tão importante quanto analisar um modo de criar ambientes, ou seja, um gênero de videogame, é mergulhar em um jogo específico e desvendar o seu funcionamento interno. A observação das constantes mudanças de jogo para jogo exige uma perspectiva macroscópica e arqueológica, que revela um movimento mais amplo de uma proposta estética e lúdica. Mas as maquinações microscópicas que operam nestes ambientes são numerosas. Se quisermos desvendar as conexões comunicativas possíveis entre estes sistemas e o jogador, elas pedem uma descrição mais cuidadosa do funcionamento interno dos ambientes.

Um conjunto de poucas regras simples como o de *Dragon Maze* atua sobre um redemoinho de inovações – algumas que sobrevivem e outras que se apagam – até desaguar em jogos com muitas regras e de extrema complexidade, que usam diferentes abordagens ao simular seus ambientes. Mesmo que estes respeitem certas premissas básicas do gênero, conseguem criar ambiências completamente distintas. Estes sistemas estabelecem um determinado vínculo sensorial com o jogador, que explora as virtualidades da simulação.

Uma ideia fundamental neste contexto é o conceito de *affordance*, cunhado por James J. Gibson (1986). Uma *affordance* é a emergência de uma propriedade nova em

um sistema a partir do encontro de dois ou mais elementos. Em *NetHack* (1987), por exemplo, o jogador se depara com alimentos que possibilitam a sobrevivência: é preciso ingerir itens comestíveis com determinada frequência, caso contrário o personagem controlado morre de fome. Estes itens possuem a *affordance*, fundamental em *NetHack*, da nutrição. Outros elementos interferirão no processo de nutrição: por exemplo, o jogador pode equipar-se com um anel mágico que desacelera seu metabolismo, exigindo menos ingestão de itens comestíveis. Isto permite a execução de mais ações que gastam energia, como correr, pular e disparar feitiços.

Esta pequena descrição da dinâmica interna de *NetHack* demonstra como os elementos (jogador, comida, equipamentos, ações) estão imbricados mutualmente: só podem ser compreendidos quando em pareamento com outros elementos. Não se sabe como um feitiço funciona até que ele seja usado contra um inimigo – e pode-se conhece-lo ainda mais se experimentado contra alvos distintos. O consumo de alimentos só age enquanto mecânica lúdica no contexto formado por outros elementos: processos que queimam energia, a necessidade de procurar comida, a iminência da morte no caso de desnutrição. O ambiente é criado no choque de agentes, que produzem as *affordances*. Ainda seguindo os rastros do pensamento de Gibson, esta criação é condicionada pela capacidade de ação de quem o navega – no caso, o jogador.

Ainda no âmbito do gênero *roguelike*, pode-se encontrar ambientes que apresentam ainda mais complexidade – fruto de um número ainda mais alto de vetores que se afetam em múltiplos contextos. *Dwarf Fortress* recusa a prevalência de espaços estreitos, e, embora possua cavernas, simula também florestas, desertos e planícies – cada lugar com seu clima, vegetação e recursos respectivos. Além de monstros, possui comunidades de trabalhadores, caravanas que viajam pelo mundo fazendo comércio e civilizações que travam guerras e disputam territórios. Se Dragon Maze dá origem a labirintos que nunca se repetem, *Dwarf Fortress* faz o mesmo com mundos inteiros.

Dwarf Fortress (2006)

O software começa simulando grandezas geográficas como a topologia do solo, a bifurcação dos rios, a presença de vegetação, o clima de cada parte do mundo, etc. Estas variáveis influenciam umas às outras: o clima afeta o tipo de vegetação, a presença de água influencia a presença da fauna, etc. Nesta dança de regras e algoritmos, é possível programar fenômenos como surgimento de desertos, vales, montanhas, florestas. O mesmo é feito com populações de seres inteligentes. Grupos de humanos, anões, elfos e *goblins* se unem em fortalezas e cidades. Cada um possui sua profissão, consome e criam produtos (a partir de recursos naturais) e possuem rotinas e até humor simulado (podem ficar satisfeitos, deprimidos ou irritados). Em grande escala, se unem em grupos maiores que podem travar guerras por territórios.

Na figura que mostra Dwarf Fortress, acima, é possível perceber que a preocupação com a dimensão visual do jogo é muito menor do que a dos processos. O jogo é apreendido através dos contextos e *affordances*, que só depois de algum tempo são assimiladas mais rapidamente pelos jogadores que dominam seu código. A mesma intensidade simulacional pode ser encontrada em *No Man's Sky* (em desenvolvimento). No entanto, desta vez ela se dá fundamentalmente de forma visual. O software do jogo simula um universo virtualmente infinito, criando de forma algorítmica desde a estrutura das galáxias até a cor da grama e das rochas de cada planeta.

No Man's Sky (2015)

Cada planeta orbita uma estrela, movimentando-se de forma elíptica ao seu redor. Dependendo da distância em relação ao astro, o planeta pode desenvolver atmosfera, que gera umidade e, possivelmente, presença de água. Água pode condicionar vida simples e, eventualmente, uma ecologia complexa, com animais e vegetais que constituem uma cadeia alimentar própria de cada mundo que se visita. Novamente, assim como os labirirntos de *Dragon Maze* e os mundos de *Dwarf Fortress*, cada sistema solar, planeta e animal em *No Man's Sky* é gerado de forma procedimental (Cf. SHAKER et al., 2015): nascem de um "esqueleto" algorítmico que define regras gerais de formação.

No caso dos animais, cada espécie (quadrupede terrestre, por exemplo) é definida por atributos: tamanho da coluna vertebral, da cabeça, da boca; tipo de dentição, presença de chifres e outras características; agressividade e posição na cadeia alimentar. Novamente, estes elementos se afetam entre si: um animal grande e com chifres provavelmente será agressivo. De um conjunto geral de regras, o computador pode gerar qualquer um dos casos específicos. Tudo depende da "semente" programada

originalmente. Em *No Man's Sky*, existem sementes que geram animais, plantas, naves, armas, rotas de comércio e exploração, entre outras. Estas sementes conversam entre si para gerar um universo dinâmico e autoorganizado.

Conexão jogador-ambiente

O ambiente que cada jogo articula a partir de suas várias camadas cria limites e aberturas para eventos lúdicos. Este labirinto de acontecimentos possíveis cria uma espécie de gramática própria de cada jogo. Um ser humano pode conectar-se a estes sistemas, atuando como "jogador", um construto que nasce do encontro de uma pessoa com uma interface (PIAS, 2011). Para se expressar dentro do ambiente, o jogador deve se familiarizar com seus limites e aberturas, o que envolve decisões lógicas, mas também (e talvez principalmente) uma modulação sensorial. Experimentando e brincando, entre sucessos e frustrações, a cognição do jogador regula-se aos poucos às regras do ambiente, corporificando sua gramática.

Esta dimensão comunicacional dos videogames, a princípio, abarca uma série de camadas que constituem o processo de jogo (NIETSCHE, 2008, p. 15-17). Entre elas, podemos destacar a produção de um imaginário simbólico do mundo ficcional; as impressões sensoriais causadas pela sua materialidade; a interface gráfica que "traduz" as maquinações do hardware para uma linguagem acessível (KITTLER, 1995); a programação e os algoritmos que controlam o software e, finalmente, a estrutura física dos circuitos elétricos e dados magnéticos. Embora estes tenham sido citados separadamente e numa ordem específica (jogador > suporte material de inputs > interface eletrônica/digital > software > hardware), todas elas se interconectam, se afetam e estabelecem fronteiras difusas umas com as outras.

Para o problema que tratamos neste artigo, não interessa, porém, selecionar jogos e explicá-los de todos estes pontos de vista. Nosso norte conceitual é outro: a ideia de que videogames podem funcionar como ambientes. Para que este modo de existência seja atualizado, ele recorre, em diferentes proporções, a cada uma das camadas de jogo citadas no parágrafo anterior. E, ainda assim, existem camadas que

chamam mais a atenção no que tange o problema de pesquisa proposto. Por exemplo, a dimensão sensorial em detrimento da simbólica. Isto porque um ambiente é uma reunião de oportunidades de vínculos entre agentes (animais), superfícies, meios e substâncias (GIBSON, 1986, p. 16-32). As vinculações que acontecem e que podem acontecer nestes jogos são, antes de mais nada, fomentadoras de sensações lúdicas: movimento, cadeias de ação-reação, controle e descontrole, etc.

Outro recorte necessário se refere ao ponto de partida escolhido para falar destes efeitos sensoriais. De fato, existe uma importante fonte de modulação e adaptação aos ambientes nos suportes materiais que os jogadores usam para enviar inputs ao sistema. Eventuais outputs podem ser sentidos através do tato, como nos controles com função vibratória, por exemplo. Mas a maior parte destes outputs é manifestada audiovisualmente através de monitores/TVs e dispositivos sonoros. Esta manifestação audiovisual adquire uma importante dimensão motora a partir do momento que se relaciona com os inputs executados pelo jogador. Este, por sua vez, passa a sentir como se controlasse o movimento de seu personagem (Cf. SWINK, 2009), e a ser afetado pela ação de agentes e eventos como paredes, buracos, explosões, aumento de velocidade e alterações (benéficas ou maléficas) nas propriedades visuais e motoras de seu *avatar*.

Desta forma, é possível perceber que a sensação de manipular um joystick (ou o arranjo teclado/mouse) associa-se fortemente às regras internas do jogo. Navegar pelas cavernas de *NetHack* implica em sensações de controle diferentes de sobrevoar estações espaciais em *No Man's Sky* – mesmo que ambos sejam jogados através do mesmo hardware. Onde, então, localiza-se o ambiente? Nos parece que ele transborda para além do próprio computador que o roda (MARTIN, 2013). No entanto, precisamos de um ponto de partida, e este será a dimensão simulacional do jogo, ou seja, a reunião de diretrizes que faz com que, quando mudemos o software que o hardware está rodando, sejamos transportados para um novo lugar com novas regras internas.

Considerações finais

Este artigo apresentou três movimentos interdependentes para lidar com uma questão única: como funciona a conexão jogador-máquina nos ambientes lúdicos do gênero *roguelike*. Nestas páginas, foi possível apresentar de maneira geral o arcabouço conceitual que permeia cada etapa, bem como as linhas gerais de como cada uma deve operar. Mas, obviamente, elas podem ser expandidas.

Na primeira parte, é necessário incorporar uma metodologia que assuma as complexidades históricas que envolvem a individuação de um modo de criação de ambientes. Preliminarmente, pode-se lançar mão das premissas da arqueologia da mídia (HUHTAMO E PARIKKA, 2011; ZIELINSKI, 2006). Esta metodologia assume como inerentes as descontinuidades envolvidas no processo de metamorfose das formas de comunicação, algo que pode ser providencial na hora de compreender a formação de um gênero de jogo.

A segunda parte imerge em jogos individuais. A hipótese aqui é que podemos tratar estes jogos não só como objetos técnicos, mas como ambientes propriamente ditos. Neste sentido, uma inspiração são os *platform studies* (MONTFORT e BOGOST, 2009). No livro *Racing the Beam*, Ian Bogost e Nick Montfort propõem uma epxloração das características do console Atari 2600 a partir da descrição de jogos e do modo como rodavam na plataforma. Sua preocupação é, sobretudo, com a materialidade do Atari. É possível deslocar este interesse para o aspecto sensorial da simulação dos videogames a partir do gênero *roguelike*. O foco muda, mas a forma de operar se assemelha.

Por último, a dimensão comunicacional destes ambientes simulados constitui o verdadeiro desafio encontrado nesta questão. Levantar as teorias que pensam a comunicação e tentar encaixá-las "à força" no fenômeno aqui apresentado pode ser um esforço pouco produtivo. Talvez uma inversão seja necessária: procurar em que instâncias do processo de jogo é possível enxergar o nascimento da comunicação. Na sua camada de aprendizado, nas modulações sensoriais exigidas pelo contato com a interface e na possibilidade de incorporar dinâmicas através da experimentação com a simulação: a ação lúdica dentro de ambientes digitais produz conexão e trocas entre jogador e máquina. Cabe a nós descobrir suas nuances.

Referências

GIBSON, James J. **The Ecological Approach to Visual Perception**. Londres: Psychology Press, 1986.

HUHTAMO, Erkki e PARIKKA, Jussi. **Media Archaeology**: Approaches, Applications, and Implications. Oakland: University of California Press, 2011.

KITTLER, Friedrich. **There is no software**. 1995. Disponível em: http://www.ctheory.net/articles.aspx?id=74. Acesso em: 2/7/2015.

MARTIN, Paul. **Landscape and gamescape in Dwarf Fortress**. The Philosophy of Computer Games Conference, 2012. Disponível em: http://www.academia.edu/4730303/Landscape_and_Gamescape_in_Dwarf_Fortress. Acesso em: 2/7/2015.

MONFORT, Nick; BOGOST, Ian. **Racing the Beam**: The Atari Video Computer System. Cambridge: The MIT Press, 2009.

NITSCHE, Michael. **Video Game Spaces**: Image, Play, and Structure in 3D Worlds Cambridge: The MIT Press, 2009.

PIAS, Claus. **The Game Player's Duty**: The User as the Gestalt of the Ports. In: HUHTAMO, Erkki e PARIKKA, Jussi. **Media Archaeology**: Approaches, Applications, and Implications. *University of California Press: 2011.*

SWINK, Steve. **Game Feel**: A Game Designer's Guide to Virtual Sensation. Amsterdam: Morgan Kaufmann/Elsevier, 2009.

TOGELIUS, Julian; SHAKER, Noor e NELSON, Mark J. **Procedural Content Generation in Games**: A Textbook and an Overview of Current Research. 2015. Disponível em: http://pcgbook.com/.

ZIELINSKI, Ziegfried. **Arqueologia da Mídia**: Em busca de um tempo remoto das técnicas do ver e do ouvir. São Paulo: Annablume, 2006.