Uma Abordagem Construcionista no Uso de Jogos Eletrônicos na Educação

Luiz M. Gerosa Davidson Cury

Universidade Federal do Espírito Santo, Departamento de Informática, Brasil

Resumo

Este artigo realiza um trabalho de investigação sobre a abordagem construcionista no uso de jogos eletrônicos na educação, abrangendo apresentar os benefícios educacionais propiciados e as principais ferramentas disponíveis.

Palavras chaves: construcionismo, jogo educacional.

Contato:
gerosa@gmail.com
dede@inf.ufes.br

1. Introdução

Nos últimos anos, conteúdos multimídias deixaram de ser criados exclusivamente por profissionais especializados e passaram a também ser criados pelos próprios consumidores, como é o caso de muitos vídeos publicados no YouTube[1] e textos sobre variados assuntos publicados em *blogs*. Essa mudança também vem acontecendo para os jogos eletrônicos através de ferramentas que possibilitam pessoas com pouco ou até mesmo nenhum conhecimento em programação criarem seus jogos.

Os jogos eletrônicos são reconhecidos por facilitar a aprendizagem de assuntos complexos e por desenvolver importantes habilidades cognitivas como, por exemplo, a resolução de problemas, a percepção, a criatividade e o raciocínio rápido [Prensky, 2001; Gee, 2003]. Quando os estudantes passam do papel de jogadores para projetistas, também há uma inversão da abordagem de aprendizagem instrucionista para construcionista [Papert, 1993], tornando os efeitos na educação ainda mais benéficos.

Outra abordagem possibilitada por essas ferramentas é os professores criarem seus próprios jogos ou simulações de modo que atendam as reais necessidades dos seus alunos [Prensky, 2006]. Desta forma, a falta de jogos prontos projetados com temática que envolva o conteúdo curricular que o professor está ministrando deixa de ser um empecilho para empregar jogos em um ambiente escolar.

O objetivo do presente trabalho é realizar uma investigação sobre a abordagem construcionista no uso de jogos eletrônicos na educação. Na Seção 2, os benefícios propiciados pela abordagem são introduzidos. Na Seção 3, as principais ferramentas disponíveis são apresentadas. Na Seção 4, o assunto é concluído.

[1] www.youtube.com

2. Benefícios Educacionais

Durante o jogo, o jogador está constantemente aprendendo as regras definidas e formulando estratégias para superar os desafios impostos. Mas pelo fato do jogador estar sujeito às regras e aos desafios criados por outras pessoas caracteriza uma abordagem de aprendizagem instrucionista. No entanto, quando as pessoas passam a criar seus próprios jogos, a abordagem de aprendizagem é construcionista [Kafai, 2001].

Em Kafai [1995], foi elaborado um estudo que colocou uma classe de crianças de dez anos de idade para fazer seus próprios jogos educacionais sobre o conceito matemático de frações. As crianças tiveram que criar seus próprios personagens, histórias e interações durante um período de seis meses. Enquanto as crianças estavam imaginando qual jogo projetar e quais características incluir, elas também estavam envolvidas em entender o que estavam aprendendo, no caso, as frações. Usando a linguagem de programação LOGO, as crianças também conseguiram aprender conceitos complexos de programação. Além disso, os estudantes lidaram com frações em seus jogos através de histórias e fantasias – contextos que são raramente promovidos em livros didáticos de matemática.

Durante o processo de criação do jogo, os estudantes aprendem como ser um bom projetista: como conceituar um projeto, como fazer uso dos recursos disponíveis, como persistir e achar alternativas para problemas inesperados e como colaborar com os outros. Em resumo, eles aprendem a gerenciar um projeto complexo do início ao fim [Resnick, 2002].

Criar jogos eletrônicos também ajuda os estudantes a desenvolver um grau maior de fluência digital. Para explicar o que é fluência digital, Resnick [2002] faz uma analogia com a aprendizagem de uma língua estrangeira. Se uma pessoa só aprende algumas frases básicas, por exemplo, para pedir orientação na rua, ela não é considerada fluente naquela língua. Para ser fluente, é necessário saber se expressar com a língua. Analogamente, a maioria das pessoas que usam os computadores não possui fluência digital. Elas sabem usar um editor de texto e navegar na Internet, mas não sabem criar algo com significado. Segundo Resnick, a

fluência digital é importante para facilitar a aprendizagem de novas tecnologias digitais, assim como saber ler e escrever fluentemente facilita a aprendizagem em vários outros assuntos.

3. Ferramentas de Criação de Jogos

Programar a lógica do jogo consiste em definir como as interfaces e as entidades reagirão aos eventos que podem ocorrer: duas entidades colidiram, a entidade foi atualizada, o jogador acionou o teclado ou o mouse, dentre outros.

As ferramentas que auxiliam a criação de jogos podem trabalhar em diferentes níveis de abstração, conforme ilustrados na Figura 1 e descrito nas Subseções 3.1 e 3.2.

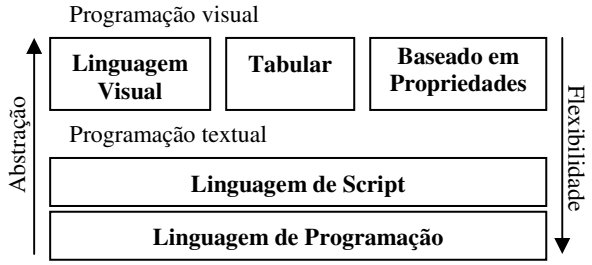

Figura 1: Diferentes níveis de abstração para criar a lógica do jogo em ferramentas de criação de jogos.

Os níveis menos abstratos usam linguagens de programação textuais, enquanto os níveis mais abstratos usam alguma forma de programação visual. É importante ressaltar que quanto maior o nível de abstração da abordagem, menor será a flexibilidade do usuário para criar seu jogo, pois alguns recursos da *engine* podem não estar disponíveis para linguagens mais abstratas.

3.1 Programação Textual

No nível menos abstrato se encontram as linguagens de programação (LP) convencionais que são usadas para criar a lógica do jogo e para acessar subsistemas como, por exemplo, gráfico, áudio, física e rede. Trabalhar nesse nível de abstração exige conhecimentos avançados de programação, aprender a usar as APIs disponibilizadas e ter conhecimentos em áreas como computação gráfica e física.

As ferramentas do segundo nível disponibilizam uma ou mais linguagens de script (LS) para acessar a *engine* do jogo. Uma LS é mais fácil de ser aprendida e usada do que uma LP convencional, pois abstraem características como gerência de memória e tipos de dados. Entretanto, para definir a lógica do jogo é necessário saber conceitos básicos de programação, aprender a sintaxe da linguagem e usar um editor de texto, o que aumenta o desafio para usuários não especialistas começarem a criar seus jogos.

Há uma grande variedade de ferramentas baseadas em LS. Por exemplo, o *Flash* [Adobe, 2007] é uma ferramenta para criação de animações 2D e pode ser usada para criar jogos usando a linguagem *ActionScript*. Outro exemplo são os ambientes que implementam a linguagem LOGO, inventada por Papert [1980], cujo propósito normalmente é educacional.

Muitos jogos comerciais liberam sua *engine* para os jogadores criarem gratuitamente variações não comerciais do jogo original, conhecidos como *Mods* [Prensky, 2006]. Eles podem alterar a aparência do jogo através de editores visuais ou alterar o comportamento do jogo através de uma LS. Muitas vezes, entretanto, só é possível usar a *engine* para criar jogos do mesmo gênero que o jogo original.

3.2 Programação Visual

Nos níveis mais abstratos, as ferramentas trabalham com algum tipo de programação visual que é usada pelos projetistas para definir a lógica do jogo. A mesma ferramenta pode disponibilizar mais de um tipo de programação visual.

Linguagem de Programação Visual

Algumas ferramentas usam uma linguagem de programação visual (LPV) que contém comandos que executam alguma ação (por exemplo, movimentar, destruir e criar entidades) e estruturas de controle: blocos condicionais e iteradores. As LPVs são orientadas a eventos, de forma que o projetista define um conjunto de comandos para os eventos que podem ocorrer durante o jogo.

O *Game Maker* [YoyoGames, 2007] é um exemplo de ferramenta que usa uma LPV. Ele possui uma versão gratuita, mas com alguns recursos bloqueados, e dispõem de uma LS para dar mais flexibilidade para os usuários avançados. Essa abordagem híbrida tem a vantagem do usuário não necessitar mudar de ferramenta caso a LPV não atenda mais. A Figura 2 mostra parte da janela de edição de comandos do *Game Maker* e foi retirada de um jogo exemplo que vem com a ferramenta. A coluna da esquerda mostra todos os eventos registrados na entidade, a coluna do meio contém os comandos registrados para o evento selecionado e a coluna da direita mostra um conjunto de comandos disponíveis na LPV.

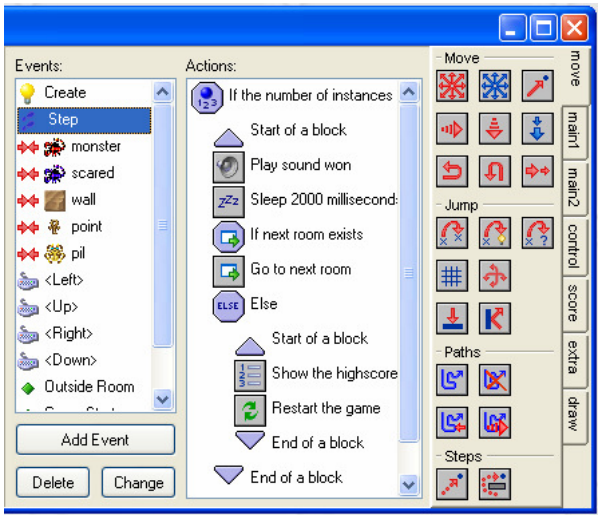

Figura 2: Janela de edição de comandos do *Game Maker*.

Outra ferramenta que usa uma LPV é o *Scratch* [MIT, 2007] (Figura 3), desenvolvido pelo *Lifelong Kindergarten Group* do MIT *Media Lab* e possui o código livre. A sua linguagem é composta por diversas estruturas de controles encontradas em LP imperativas e o formato de blocos que se encaixam (Figura 4) facilita a aprendizagem e o seu uso.

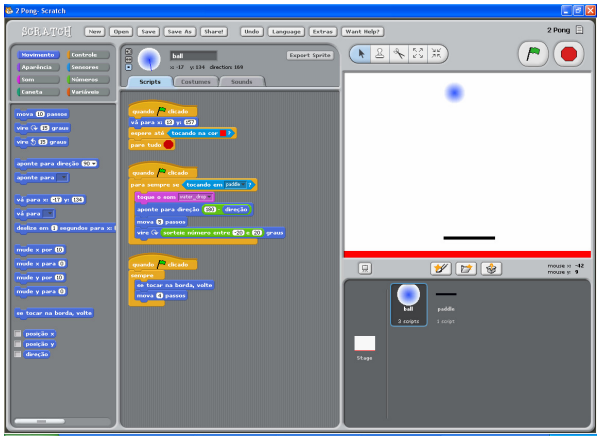

Figura 3: Imagem da ferramenta *Scratch*

Figura 4: LPV usada no *Scratch*

Um diferencial do *Scratch* são os recursos voltados ao compartilhamento e avaliação dos jogos criados pelos usuários através de uma comunidade *online* [Monroy-Hernandez, 2007].

Programação Tabular

O *Klik & Play* [Clickteam, 1994] foi uma das primeiras ferramentas com o propósito de ser usado por não especialistas para criar jogos. Ele fez muito sucesso no Brasil e é gratuito para fins educacionais. O *Klik & Play* evoluiu e mudou de nome algumas vezes, sendo que a última versão se chama *The Games Factory 2*, específica para criar jogos ou *Multimedia Fusion 2*, para jogos e aplicativos [Clickteam, 2006]. Essa linha de ferramentas usa uma forma de programação tabular, que apesar de não ser uma LP, pode ser usada para introduzir conceitos algorítmicos. A Figura 5 mostra o editor de eventos da ferramenta, onde o projetista consegue visualizar todas as interações entre as entidades e entre o jogador e o jogo.

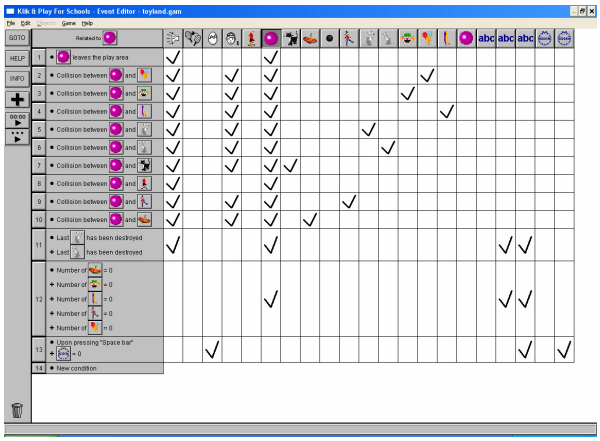

Figura 5: Editor de eventos do *Klik & Play*.

Programação Baseada em Propriedades

Na programação baseada em propriedades, a lógica do jogo é definida através de uma interface de configuração de propriedades. Ao invés de programar um comportamento, o projetista deve configurar o comportamento nas entidades criadas. Por exemplo, o *Klik & Play*, citado anteriormente, usa esse tipo de programação para definir qual o tipo de movimentação da entidade: ser controlada pelo teclado ou mouse, movimentar-se aleatoriamente, seguir um caminho especificado, dentre outros.

A vantagem de usar propriedades é que o processo de criação se torna muito mais simples para os usuários que nunca programaram, pois diversos controles que seriam necessários são abstraídos. As desvantagens são a redução da capacidade da ferramenta desenvolver a habilidade de programação nos usuários e a redução da flexibilidade dos usuários criarem novos comportamentos. A falta de flexibilidade normalmente é compensada através de *plug-ins* e de uma linguagem de script que são usados para estender as funcionalidades da ferramenta e criar a lógica específica do jogo.

O *Torque Game Builder* [GarageGames, 2007] é um exemplo de ferramenta comercial que usa uma programação baseada em propriedades. Ele possui um sistema de componentes que são anexados às entidades para definir o comportamento e uma linguagem de

script que pode ser usada, opcionalmente, para criar comportamentos não disponíveis nos componentes.

4. Conclusão

A abordagem construcionista no uso de jogos eletrônicos na educação se mostrou bastante promissora para desenvolver importantes habilidades nos estudantes e, ao mesmo tempo, tornar a aprendizagem mais lúdica.

A escolha da ferramenta a ser usada pelos alunos e professores deve levar em consideração o propósito educacional para a ferramenta. Caso o propósito educacional seja desenvolver a habilidade de programação e o raciocínio lógico, é mais apropriado escolher uma ferramenta que trabalhe com uma linguagem de programação visual. Caso o propósito seja usar a ferramenta para a construção do conhecimento em outras áreas, é mais apropriado escolher uma ferramenta que trabalhe com programação tabular ou baseada em propriedades, pois as dificuldades dos alunos serão menores e eles estarão mais engajados em aprender a matéria desejada.

Além do propósito educacional, é importante observar nas ferramentas características como:

- Usabilidade: observar se a ferramenta tem uma interface fácil de ser usada e aprendida.
- Recursos: observer a abrangência dos recursos disponíveis na ferramenta para criar o jogo.
- Flexibilidade: observar se a ferramenta possibilita usuários avançados importarem novos recursos e usarem uma linguagem de script.
- Compartilhamento: observar se a ferramenta disponibiliza uma comunidade de usuários onde os estudantes possam compartilhar seus projetos e se a ferramenta oferece recursos para publicar e reusar outros projetos no jogo que está sendo criado.

Referências

Adobe (2007) "*Flash*", disponível em <www.adobe.com/products/flash>, acesso em 28/08/2007.

Clickteam (1994), "*Klik & Play*", disponível em <members.fortunecity.com/oponto/klik.html>, acesso em 28/08/2007.

Clickteam (1996), "*The Games Factory 2*" e "*Multimedia Fusion 2*", disponíveis em <www.clickteam.com>, acesso em 28/08/07.

GarageGames (2007), "*Torque Game Builder*", disponível em <www.garagegames.com/products/torque/tgb>, último acesso em 28/08/2007.

Gee, J.P. (2003), *"What Video Games Have to Teach Us About Learning and Literacy"*, Palgrave Macmillan.

Kafai, Y. B. (1995), *"Minds in play: Computer game design as a context for children's learning"*. Hillsdale, NJ: Erlbaum.

Kafai, Y. B. (2001). *"The educational potential of electronic games: From games–to–teach to games–to–learn"*. Chicago: Playing by the Rules Cultural Policy Center, University of Chicago.

MIT (2007), "*Scratch*", disponível em <scratch.mit.edu>, acesso em 28/08/2007.

Monroy-Hernandez, A. (2007) *"ScratchR: sharing user-generated programmable media"*, Interaction Design for Children Conference, Denmark.

Papert, S. (1980). *"Midstrorms: Children, computers and powerful ideas"*. NY: Basic Books.

Papert, S. (1993). *"The Children's Machine"*. New York: Basic Books.

Prensky, M. (2001). *"Digital Game-Based Learning"*. New York: McGraw Hill.

Prensky, M. (2006). *"Don't bother me mom, I'm learning!"* Paragon House Publishers.

Resnick, M. (2002). *"Rethinking learning in the digital age"*. In G. Kirkman (Ed.). The global information technology report: Readiness for the networked world. Oxford: Oxford University Press.

YoyoGames (2007), "*Game Maker*", disponível em <www.yoyogames.com>, acesso em 27/08/07.