Testando a Diversão em um Jogo Sério para o Aprendizado Introdutório de Programação

Adilson Vahldick[1, 2], António José Mendes[1], Maria José Marcelino[1], Maciel Hogenn[2], Pablo Schoeffel[2]

[1]Departamento de Engenharia Informática – Universidade de Coimbra – Portugal

[2]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC) – Ibirama, SC – Brasil

{toze, zemar}@dei.uc.pt, {adilson.vahldick, pablo.schoeffel}@udesc.br, maciel.hog@gmail.com

Abstract. *This article describes a serious game for learning introductory programming and its experimentation with a class of 16 students. The game covers three subjects: action sequence, variables and conditional. The solution is developed by visual blocks as in Scratch. The goal of the experiment was to identify the weak points of the game to promote enjoyment. EGameFlow was the instrument used to measure this quality in six dimensions: concentration, goal clarity, feedback, challenges, autonomy and immersion. The results showed that the immersion is the weakest dimension and, in the conclusion, we present suggestions for its improvement.*

Resumo. *Este artigo descreve um jogo sério para o aprendizado introdutório de programação e sua experimentação com uma turma de 16 alunos. O jogo abrange os assuntos de sequenciamento de ações, variáveis e condicionais. A solução é desenvolvida com blocos gráficos, tal como no Scratch. O objetivo da experimentação foi identificar os pontos fracos do jogo em promover a diversão. O instrumento utilizado para aferir essa qualidade chama-se EGameFlow que permitiu medí-lo em seis dimensões: concentração, clareza dos objetivos, feedback, desafios, autonomia e imersão. Os resultados demonstraram que a imersão é a dimensão mais fraca e nas conclusões apresentamos sugestões de melhorias.*

1. Introdução

Aprender programação de computadores não é simples para muitos estudantes. Alguns fatores relevantes podem influenciar a apropriação de competências necessárias, tais como motivação, persistência, confiança, responsabilidade emocional, estratégias de resolução de problemas (Winslow, 1996; Gomes & Mendes, 2007). Além disso, os alunos devem praticar intensivamente (Robins et al., 2003) e estar motivados para resolver vários exercícios. Algumas universidades fomentam o desenvolvimento de pequenos jogos como uma alternativa aos exercícios tradicionais de programação (Barnes et al., 2007; Morrison & Preston, 2009). Outras utilizam jogos para auxiliar no entendimento de conceitos abstratos, para diminuir o tempo entre a teoria e prática, reforçar assuntos e conteúdos, e como alternativa aos exercícios comuns e tarefas repetitivas (Eagle & Barnes, 2009).

Esses jogos desenvolvidos primariamente para atender objetivos educacionais, que são nomeados de "jogos sérios", não têm o entretenimento como propósito principal. Porém, para seu sucesso, como qualquer jogo comercial, dependem da coerente aplicação de regras, mecânicas e elementos de jogos como, por exemplo, enredo, pontuação, missões, interação entre personagens (Zyda, 2005; Kapp, 2012). De qualquer forma, a diversão é o objetivo mais importante a ser alcançado com um jogo (Koster, 2014). De contrário, pode-se tornar aborrecido, o jogador pode perder a motivação e não se concentrar na tarefa, assim não atingir os objetivos de aprendizagem. A diversão permite que o jogador realize as suas tarefas de forma mais fácil, mesmo quando exigem mais esforço (Prensky, 2001). O sucesso na experiência de jogar está relacionado com o grau de comprometimento da atenção e imaginação do jogador (Gee, 2004). Esse comprometimento é resultado de um fluxo e sequência de eventos que mantém o jogador interessado (Schell, 2008) e a sua manutenção depende dos desafios, curiosidade e fantasia promovidos pelo jogo (Malone, 1980).

Vahldick et al. (2014) analisaram e classificaram 40 jogos que podem ser usados no aprendizado de programação. Aproveitando as lacunas identificadas naquele trabalho, neste artigo é apresentado um novo jogo sério para o aprendizado de programação, chamado NoBug's Snack Bar. Outro ponto observado em (Vahldick et al., 2014) é que os jogos relacionados com algum tipo de publicação em conferências e periódicos carecem de aferição quanto à motivação intrínseca no divertimento do jogador. Assim, o jogo aqui apresentado é validado através do prazer em jogar utilizando o instrumento EGameFlow (Fu et al., 2009). Neste trabalho tenta-se identificar elementos e mecânicas que possam tornar o jogo divertido, e posteriormente ajustá-lo para promover a eficiência na aprendizagem autônoma com o jogo.

Este artigo está organizado da seguinte forma: a seção 2 apresenta os trabalhos correlatos e alerta para a diferença entre os jogos sérios para o aprendizado de programação e dos ambientes lúdicos para programação. Na seção 3 é descrito o jogo e na seção 4 o experimento. A seção 5 apresenta os resultados e a sua discussão e a última seção apresenta as conclusões e as ideias para os trabalhos futuros.

2. Trabalhos Correlatos

Os jogos sérios referidos por Vahldick et al. (2014) foram divididos em três tipos: LOGO-like, jogos de aventura e desafios em geral. LOGO-like são jogos de ação em que o jogador programa os movimentos de um robô, de uma tartaruga ou de outro tipo de personagem num mundo virtual. As missões são para alcançar uma posição no mundo ou coletar uma quantidade de objetos. Os movimentos são programados com linguagens de programação simplificadas. Em todos os jogos deste tipo, os programas são criados através de arrastar-e-soltar comandos (blocos visuais) de uma barra de botões. Isso elimina os erros de sintaxe. Nos jogos de aventura, o jogador comanda um herói que explora um mundo, coleta objetos e interage com outros personagens controlados pelo jogo. Os demais jogos encontrados pertencem a vários tipos, como simuladores, de estratégia em tempo real e labirintos, e os autores classificaram-nos todos num único grupo denominado desafios em geral.

Os jogos do tipo LOGO-Like parecem mais promissores, pois usam uma abordagem mais prática solicitando que o jogador resolva pequenas e evolutivas missões repetidamente. As novas habilidades são adquiridas quando são requeridas e praticadas várias vezes durante a experiência de jogo (Kapp, 2012). No aprendizado de programação, os alunos adquirem habilidades através da constante prática e avaliação dos

seus resultados (Perkins & Martin, 1986; Prince & Hoyt, 2002). O jogador tem um pequeno conjunto de objetivos em cada missão e tenta cumprí-los em poucos minutos. Assim, pratica diversas vezes o mesmo conceito e aprende um novo conceito em cada nova missão. A avaliação é intrínseca neste tipo de jogo através do sucesso ou falha na missão. Quando o jogador não completa a missão, o jogo apresenta comentários informando-o do que fez de errado, e talvez algumas dicas para vencer a missão.

Três jogos do tipo LOGO-Like serviram de inspiração a este jogo: Code.org, Robozzle[1] e Gidget (Lee et al., 2014). Code.org é um esforço recente do governo norte-americano para promover o ensino de computação entre as crianças. O jogo cobre um grande conjunto de tópicos, da execução passo-a-passo até chamadas de funções e passagem de parâmetros. O jogador usa os blocos visuais de forma similar ao Scratch. Robozzle é um ambiente que promove uma comunidade de jogadores. Permite a criação de desafios e os usuários conseguem partilhá-los entre si. O ambiente gerencia os recordes de cada missão e as melhores posições no geral (somando todas as missões). Diferente dos demais jogos LOGO-Like, no Gidget o personagem não anda um passo de cada vez, mas vai diretamente ao objeto ou outros personagens no jogo.

3. O Jogo NoBugs' Snack Bar

A mecânica do jogo NoBugs' Snack Bar é inspirada nos jogos de gestão de tempo. Neste tipo de jogo, as tarefas são divididas em passos sequenciais em que o jogador precisa cumprí-las dentro de um limite de tempo (Trefry, 2010). O contexto do jogo é baseado nas experiências cotidianas, como um restaurante, uma loja de roupas ou uma barbearia. O jogador controla o funcionário do estabelecimento e o jogo controla os clientes que efetuam pedidos. O progresso do jogo é baseado em atender a maior quantidade de pedidos dentro de um tempo limite. Em cada nova missão, os clientes fazem pedidos mais complexos, que requerem novas e diferentes sequências de passos e combinações de itens. A combinação de um ou mais elementos para criar um novo item requer habilidades de resolução de problemas (Kapp, 2012). No jogo descrito nesse trabalho, o jogador controla o atendente de uma lanchonete e os clientes solicitam combinações de comidas e bebidas. O atendente precisa ir às máquinas e equipamentos onde essas são preparadas ou estão disponíveis. A missão termina quando o jogador atender os pedidos de todos os clientes. Algumas missões são limitadas no tempo, outras na quantidade de código e outras na quantidade de variáveis usadas. O código é produzido através de arrastar-e-soltar blocos visuais como no Scratch. Foram definidos um conjunto inicial de regras e elementos de jogo:

- O apetite dos clientes, ou seja, as bebidas (refrigerante ou suco) e comida (cachorro-quente), e consequentemente os locais onde os pedidos são preparados (mostrador-quente, geladeira, caixa e espremedor de frutas);
- Os objetivos das missões são definidos por esses apetites. É também possível incluir algumas limitações nos objetivos, como a quantidade de blocos e/ou a quantidade de variáveis que podem ser usadas no programa;
- O sistema de pontuação: a maneira como os jogadores conquistam pontos após completarem os objetivos, incluindo a definição de bônus baseados no tempo em que foi gasto para resolver a missão;
- Textos explicativos e o enredo da missão;

[1] http://robozzle.com

- Código inicial da missão: a missão pode iniciar com alguns blocos predefinidos ou reusar o código do próprio jogador em missões anteriores.

Existe um conjunto de comandos para o atendente realizar as tarefas, associados de acordo com o tipo de apetite do cliente. A Figura 1a apresenta um trecho da solução da missão 15 e a Figura 1b apresenta a área da lanchonete. Essa missão tem como descrição: *Agora os nossos clientes podem ter sede OU fome. E, se têm sede, podem pedir refrigerante ou sumo. Para resolver essa missão precisas usar um bloco de condicional dentro de outro.* O primeiro bloco serve para o atendente andar até a posição 1 do balcão do bar (representado por A na Figura 1b). Então pergunta ao cliente se tem sede. Em caso afirmativo, pergunta o que deseja beber e guarda o pedido na variável *pedido.* Em seguida o programa verifica se o pedido é um refrigerante. Em caso afirmativo, o atendente vai até a geladeira (B na Figura 1b), pega a bebida de acordo com o *pedido* e armazena na variável *pedido*. Caso contrário, vai até a caixa de frutas (C na Figura 1b), pega a fruta de acordo com o *pedido* e armazena na variável *frutas*. Em seguida, vai até a máquina de frutas (D na Figura 1b), prepara o suco e armazena na variável *pedido*. Caso o cliente não tivesse sede, então entramos num bloco para atender a fome. O atendente pergunta ao cliente o que ele quer comer e armazena em *pedido*. Depois vai ao mostrador-quente, pega o cachorro-quente de acordo com o *pedido* e armazena na variável *pedido.* Após finalizar esse bloco condicional mais externo, o atendente volta até o cliente e entrega o que ele pediu. Nesse momento, se a entrega combina com o pedido, o jogador recebe dinheiro, e provavelmente cumpre uma das metas da missão. Caso contrário, o cliente fica com expressão de raiva, o jogador recebe uma mensagem de erro com uma explicação do problema e a execução termina.

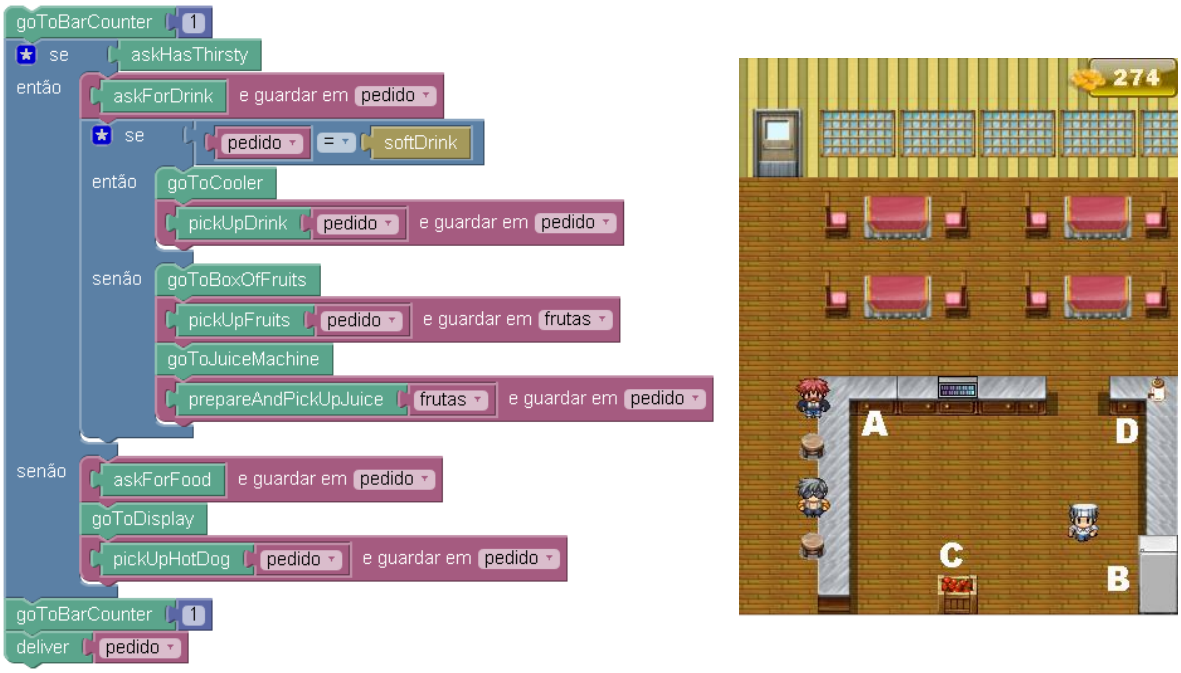

Figura 1. (a) Blocos de código para missão 15; (b) a área da lanchonete

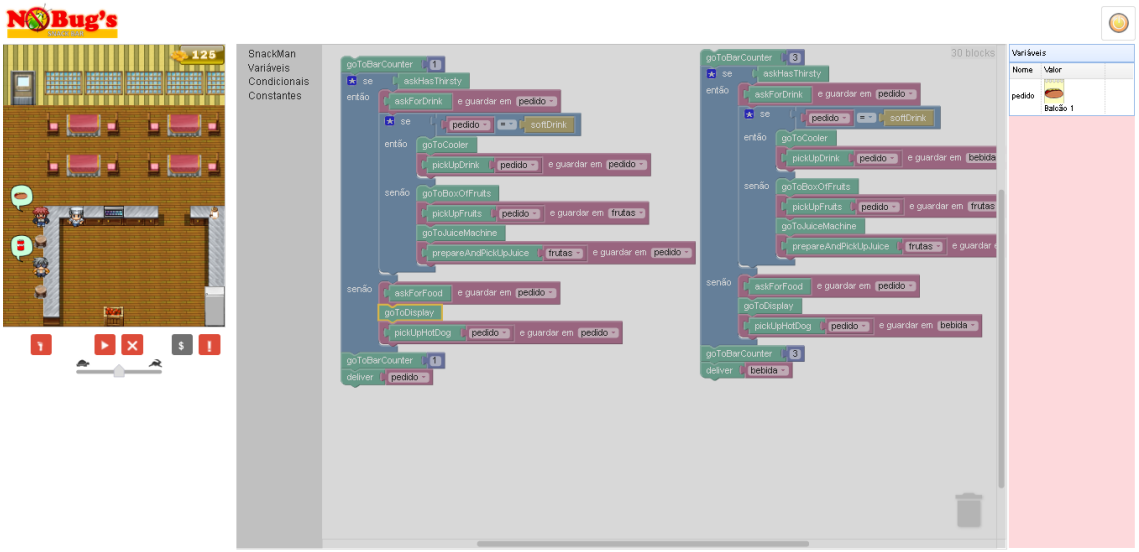

Figura 2. Ambiente do jogo

Foi definido um conjunto de 16 missões cobrindo tópicos básicos, como manipulação de variáveis, sequenciamento de ações e condicionais. O jogo está dividido em duas fases: na primeira estão as missões referentes à manipulação de variáveis e sequenciamento das ações; na segunda estão os condicionais, representando a capacidade do personagem tomar decisões. As missões foram organizadas da seguinte forma:

- Missões 1 a 3: o jogador aprende o essencial de arrastar e unir os blocos de comandos, executar e depurar o programa;
- Missões 4 a 6: o jogador aprende manipulação de variáveis e sequência de ações;
- Missão 7: as mesmas metas da missão 6, mas limitadas na quantidade de blocos;
- Missão 8: mais complexidade na sequência de ações;
- Missões 9 e 10: o jogador recebe bônus se resolve o problema dentro de um intervalo de tempo, e tem restrições quanto à quantidade de variáveis;
- Missão 11 a 16: uso de condicionais, iniciando com condicionais simples, seguindo com a alternativa "senão" e finalizando com condicionais aninhados.

O jogo foi desenvolvido em HTML5, Java e o framework Blockly[2], e executa num navegador comum. A Figura 2 ilustra o ambiente do jogo: à esquerda está a animação e a área com os botões para depurar, executar, parar a execução, equipar a lanchonete e consultar os objetivos da missão; a área de codificação está ao centro; e a lista de variáveis à direita. O jogador consegue executar ou depurar o programa. Na depuração, o jogo apresenta uma lista de variáveis e executa um passo de cada vez após cada clique no botão depurar.

4. Experimentação

O objetivo desta experiência foi mensurar o prazer dos alunos com o jogo e identificar lacunas para melhorar a sua diversão. O teste foi conduzido com 16 alunos de primeiro semestre do Bacharelado de Engenharia de Software da UDESC após a segunda semana de aula. Essa quantidade foi limitada aos alunos que compareceram na aula em que foi feito o experimento. No momento da aplicação, os alunos já tiveram sete aulas com

[2] https://developers.google.com/blockly/

assuntos sobre variáveis, expressões lógicas e condicionais. O experimento aconteceu durante uma aula (1,5 horas). Primeiramente, os alunos responderam um questionário demográfico e a outro para classificar o jogador através do coeficiente de Bartle (Bartle, 2003). Em seguida, jogaram livremente por 1 hora. No final, responderam um questionário adaptado do modelo EGameFlow (Fu et al., 2009) para avaliar a diversão no jogo.

Bartle (2003) classifica o comportamento dos jogadores em 4 tipos: socializadores (gostam de interagir com outros jogadores), exploradores (gostam de descobrir novos itens, cenários e personagens), empreendedores (gostam de interagir no mundo, conquistando objetivos, pontos e reputação) e assassinos (gostam de interagir nos outros jogadores, ou seja, mudando o estado deles, por exemplo eliminando-os ou promovendo-os). Esta classificação auxilia os projetistas de jogos a determinarem elementos que divertem os tipos de jogadores que desejam atingir. Entretanto, um jogador não é totalmente de um tipo ou de outro, mas diverte-se com mais ou menos intensidade em cada um dos estilos. Dado o pequeno tamanho da população, neste estudo utilizamos apenas os dois principais estilos de cada jogador para ordenar a turma.

EGameFlow (Fu et al., 2009) é um instrumento composto de um conjunto de escalas que avaliam a satisfação dos usuários nos jogos sérios baseado em oito componentes de um fenômeno psicológico chamado "fluxo" (Csikszentmihalyi, 1990): concentração, desafio, habilidades (substituído no EGameFlow por aperfeiçoamento no conhecimento), controle, clareza dos objetivos, *feedback*, imersão e interação social. Fluxo é o estado em que uma pessoa está tão envolvida com uma atividade que nada mais importa ao seu redor. EGameFlow foi uma adaptação para jogos sérios de GameFlow (Sweetser & Wyeth, 2005). Este trabalho adaptou o EGameFlow desconsiderando as dimensões interação social, pois não é um jogo *multiplayer*, e o aperfeiçoamento no conhecimento, pois, apesar de ser um jogo sério, neste momento, o foco está em avaliar a diversão. O questionário utilizado foi composto por 29 perguntas com respostas numa escala Likert de 1 (Discordo plenamente) a 5 (Concordo plenamente). Selecionamos o EGameFlow em vez do GameFlow, pois as dimensões que utilizamos também foram adaptadas entre as duas versões.

5. Resultados e Discussão

A Tabela 1 mostra os dados demográficos dos participantes. Além do sexo e idade, também foram perguntados com que frequência que ele joga, quantas horas joga em cada seção (2,63 horas ± 1,63), desse tempo quanto é dedicado a jogar *online* com outros jogadores, quantos anos joga (8,38 anos ± 3,24) e seu conhecimento prévio em programação ou algoritmos. Pode-se observar que a maioria (58,8%) tem o hábito de jogar diária ou semanalmente e 62,4% responderam que não costumam jogar *online* com outros jogadores. Esta população que gosta de jogar sozinha, combina com o tipo de jogo do NoBug's SnackBar.

Tabela 1. Dados demográficos dos participantes

	Sexo		Idade			Frequência que joga				Multiplayer				Conhecimento prévio			
	M	F	<=17	18 ou 19	>=20	Diariamente	Semanalmente	Mensalmente	Não tem hábito de jogar	Essencialmente	Metade do tempo	Pouco	Não tem hábito de jogar	Nenhum	Fiz umas cadeiras	Fiz um curso técn./superior	Trabalho com programação
Indiv. (%)	15 (93,8)	1 (6,3)	9 (56,3)	3 (18,8)	4 (25,0)	4 (25,0)	7 (43,8)	3 (18,8)	2 (12,4)	3 (18,8)	3 (18,8)	6 (37,4)	4 (25,0)	8 (50,0)	6 (37,4)	1 (6,3)	1 (6,3)

A Tabela 2 apresenta a distribuição de frequências dos alunos conforme o coeficiente de Bartle[3]. Pode-se observar que a maioria (74,8%) tem o tipo Assassino (K) como principal comportamento de jogador. Como não é um jogo violento, isso pode vir a estar relacionado com uma possível avaliação não positiva na satisfação do jogo.

Tabela 2. Coeficiente de Bartle dos participantes

	AE	EA	EK	ES	KA	KE
Indivíduos (%)	1 (6,3)	1 (6,3)	1 (6,3)	1 (6,3)	5 (31,3)	7 (43,5)

A Figura 3 apresenta o quociente entre o tempo total gasto com a missão e a quantidade de tentativas, ou seja, é o tempo médio que o aluno utilizou para pensar e resolver entre cada um das tentativas dentro da missão. Excluíram-se as três missões iniciais, pois são direcionadas para ambientação aos recursos do jogo. Os desafios começam a partir da quarta missão. Entre parêntesis está a quantidade de alunos que concluíram cada missão dentro do limite de 1 hora da aula destinada ao jogo. Em uma hora de jogo 3 alunos concluíram 12 missões, por isso as missões 13 a 16 não estão inclusas na análise. A partir desse gráfico pode-se analisar como uma curva de interesse (Schell, 2008), em que as medianas podem ser lidas como os pontos da curva. Se a mediana de uma missão for maior que a mediana da missão anterior, representa que a missão anterior foi mais fácil que a seguinte. Schell (2008) sugere que haja momentos de dificuldade (missões 6, 9 e 12) e outros em que o jogador sinta-se no controle do jogo.

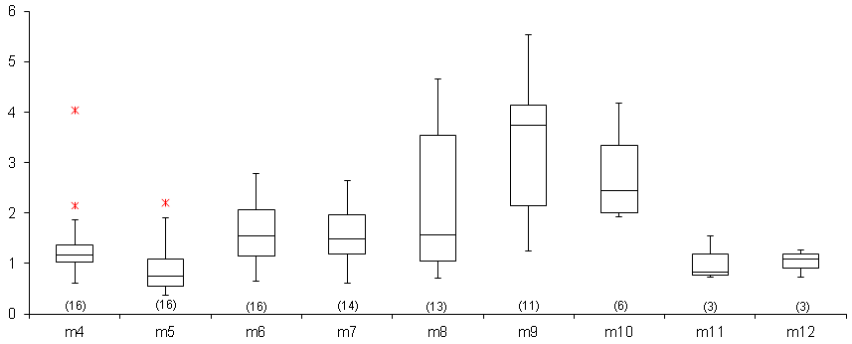

Figura 3. Tempo médio (minutos) gasto por missão

A Tabela 3 representa as médias das respostas do questionário EGameFlow. A consistência interna do questionário foi avaliada por meio do coeficiente alfa de Cronbach (α=0,916). O coeficiente de correlação de Spearman foi utilizado para estudar as relações entre as variáveis: escala do EGameFlow, o tempo médio gasto por missão e o tipo do jogador. Foi observada somente uma correlação moderada significante (r=0,526, ρ=0,036 < α=0,05) entre o tempo médio por missão com a dimensão **desafio** do EGameFlow. Para as demais dimensões, e relações entre as variáveis, as correlações foram fracas e não significantes.

Tabela 3. Escala EGameFlow

	Concentração	Clareza nos objetivos	Feedback	Desafios	Autonomia	Imersão
Média (DV)	4,1 (0,46)	4,4 (0,58)	4,3 (0,57)	4,3 (0,61)	4,2 (0,63)	3,6 (0,56)

[3] O original em inglês é **S**ocializer, **E**xplorer, **A**chiever e **K**iller. Utilizaremos as iniciais em inglês para classificação dos jogadores.

Transformando cada média numa proporção entre 0 a 100, todas as dimensões apresentaram médias acima dos 70 (p.e., 3,6 →72). Em relação à dimensão imersão, aquela com menor média, podia-se supor que fosse esperado visto que, segundo a classificação de Bartle, a maior parte dos alunos tinha Assassino como estilo de jogo primário. Empregou-se o teste T para grupos independentes, para testar a hipótese nula de que a *avaliação da imersão foi igual entre os Assassinos e os demais*. Através dos resultados ($t=0,635$; $p=0,536 > \alpha=0,05$) não pode-se rejeitar a hipótese nula. Assim, conclui-se que a turma toda apresentou o mesmo sentimento em relação à imersão.

Após responderem ao questionário EGameFlow, os alunos ainda tinham duas questões para indicarem o que mais e o que menos gostaram no jogo. Notadamente, o fato de haver um jogo em que pudessem desenvolver e serem testados quanto à lógica e ao raciocínio de programação foi o ponto forte da maioria dos alunos. Quanto aos pontos fracos, apesar de Squire (2005) mencionar que os alunos se preocupam mais com os elementos do jogo do que a sua qualidade gráfica, houveram três alunos que apontaram descontentamento quando à estética. Outros dois sentiram falta de mais explicações e dicas durante o jogo e um deles sugeriu que o jogo fornecesse a solução da missão quando não conseguisse alcançá-la.

A imersão refere-se ao nível de envolvimento, engajamento e concentração dos jogadores (Brown & Cairns, 2004). De acordo com Sweetser & Johnson (2004), dois elementos que podem auxiliar no processo de imersão é o áudio (efeitos sonoros e trilha sonora) e a narrativa (introdução e enredo). No momento, o jogo não tem recursos de áudio. Uma possível melhoria é a introdução de som ambiente e os clientes e atendente emitirem alguns sons. Quanto à narrativa, as missões e objetivos do jogo podem estar mais envolvidos com a estória.

Clareza nos objetivos, Feedback e Desafios foram as dimensões melhor avaliadas. Quanto à clareza não existem respostas nas questões abertas. Para o feedback existem três respostas negativas em relação à explicação dos erros no seus programas. Logo, precisamos rever o suporte quando os alunos falham no cumprimento de algum dos objetivos da missão. Em relação aos desafios pode-se utilizar a análise da curva de interesse mencionada sobre a Figura 3. Verificou-se que existe uma variação nos momentos de dificuldades entre as missões, o que favorece o engajamento. Entretanto, as missões 6 e 7 estão no mesmo nível de dificuldade. Na missão 7 o jogador precisa reescrever o seu programa da missão 6 otimizando o código, diminuindo o programa em um bloco. Como a natureza da tarefa é diferente entre ambas as missões, na missão 6 cria a solução, e na missão 7 otimiza a mesma solução, acreditamos não haver a necessidade de modificações.

6. Conclusões

Apesar dos jogos já serem usados no aprendizado da programação, ainda existem algumas lacunas que podem ser exploradas para o desenvolvimento de novos jogos. O jogo NoBug´s SnackBar abrange os assuntos iniciais de qualquer disciplina de programação e o presente artigo relatou um primeiro experimento do jogo com alunos. O objetivo foi verificar o quanto as decisões tomadas pelos investigadores na escolha da mecânica e elementos do jogo precisam ou não serem ajustadas para a diversão. Quanto ao conteúdo deseja-se que o jogo abranja todos os assuntos de uma disciplina introdutória. Mas antes de evoluí-lo no desenvolvimento, foi preciso averiguar se o jogo pode ser considerado divertido e motivador. Para essa avaliação considerou-se utilizar um instrumento chamado EGameFlow e também com os comentários dos alunos. Essas

respostas apontaram a imersão como uma dimensão que precisa de melhorias. A média geral foi 4.1, transformando numa escala de 0-100, sua nota foi 82. Consideramos uma excelente nota, e apesar das melhorias necessárias na imersão, podemos prosseguir o trabalho de desenvolvimento e pesquisa com mecânica e elementos citados na seção 3.

Nossa análise e experimento estão sujeitas a algumas limitações. O experimento foi conduzido no período de uma aula. Todos os alunos consumiram o mesmo tempo para jogar. Entretanto, como pudemos observar, eles não jogaram a mesma quantidade de missões. As pessoas exigem tempos diferentes para se satisfazerem. Provavelmente poderíamos mensurar melhor a diversão se o fizessemos uma segunda vez a todos após finalizarem a última missão. Assim também seria possível comparar os dois momentos e verificar mudanças no nível de diversão. Contudo, isso exigiria mais do que uma aula ou solicitar aos alunos jogarem fora de aula. A primeira opção não foi possível pois comprometeria o trabalho do professor e a segunda opção seria justificada se o jogo estivesse integrado à própria disciplina. Na sequência dessa investigação, estamos planejando usar o jogo durante todo um semestre, e assim avaliar com mais precisão a motivação e as influências quanto ao comportamento e aprimoramento no conhecimento dos alunos.

Como trabalho futuro, além do que já foi mencionado (novas missões para cobrir os demais assuntos da disciplina e melhorias para a imersão do jogador), também está a ser desenvolvido o jogo de forma que o próprio professor possa especificar as missões, por exemplo, as metas da missão, a dinâmica dos clientes, e os comandos disponíveis. Acredita-se que esses recursos venham a facilitar a adoção do jogo para uma quantidade maior de professores e turmas.

Agradecimentos

AV agradece a bolsa de doutorado apoiada pelo CNPq/CAPES – Programa Ciência sem Fronteiras – CsF (6392-13-0) e autorização de afastamento da UDESC (688/13). MH agradece a bolsa de mobilidade estudantil apoiada pela UDESC – PROME (01/2014). Os autores também agradecem a disponibilidade de professor e alunos envolvidos no experimento.

Referências

Barnes, T., Powell, E., Chaffin, A., Godwin, A., & Richter, H. (2007). Game2Learn: Building CS1 Learning Games for Retention. In *12th SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 121–125). Dundee, Scotland.

Bartle, R. A. (2003). *Designing Virtual Worlds*. New Riders Publishing.

Brown, E., & Cairns, P. (2004). A grounded investigation of game immersion. In *CHI '04 extended abstracts on Human factors in computing systems* (pp. 1297–1300). Vienna, Austria.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. New York: Harper Perennial.

Eagle, M., & Barnes, T. (2009). Experimental Evaluation of an Educational Game for Improved Learning in Introductory Computing. In *40th ACM Technical Symposium on Computer Science Education* (pp. 321–325). Chattanooga, USA: ACM Press.

Fu, F. L., Su, R. C., & Yu, S. C. (2009). EGameFlow: A scale to measure learners' enjoyment of e-learning games. *Computers and Education*, *52*(1), 101–112.

Gee, J. P. (2004). Learning by design: Games as learning machines. *Interactive Educational Multimedia*, *8*(8), 15–23.

Gomes, A., & Mendes, A. J. N. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering Education* (pp. 1–5). Coimbra, Portugal.

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. San Francisco, CA: Pfeiffer.

Koster, R. (2014). *A Theory of Fun for Game Design* (2nd ed.). O' Reilly Media, Inc.

Lee, M. J., Bahmani, F., Kwan, I., Laferte, J., Charters, P., Horvath, A., … Ko, A. J. (2014). Principles of a Debugging-First Puzzle Game for Computing Education. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 57–64).

Malone, T. W. (1980). What makes things fun to learn? Heuristics for designing instructional computer games. In *3rd ACM SIGSMALL Symposium* (Vol. 162, pp. 162–169). Palo Alto, California, USA.

Morrison, B. B., & Preston, J. A. (2009). Engagement : Gaming throughout the Curriculum. In *40th ACM Technical Symposium on Computer Science Education* (pp. 342–346). Chattanooga, USA.

Perkins, D., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In E. Soloway and S. Iyengar (Ed.), *Empirical studies of programmers* (pp. 213–229). Ablex, NJ.

Prensky, M. (2001). *Digital Game-Based Learning*. McGraw-Hill.

Prince, M., & Hoyt, B. (2002). Helping students make the transition from novice to expert problem-solvers. In *32nd Frontiers in Education, 2002.* (pp. 7–11).

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172.

Schell, J. (2008). *The Art of Game Design: A book of lenses*. Burlington, MA: Elsevier Inc.

Squire, K. D. (2005). Resuscitating research in educational technology: Using game-based learning research as a lens for looking at design-based research. *Educational Technology*, *45*(1), 8–14.

Sweetser, P., & Johnson, D. (2004). Player-Centered Game Environments: Assessing Player Opinions, Experiences, and Issues. In *Entertainment Computing – ICEC 2004* (Vol. LNCS 3166, pp. 321–332). New York.

Sweetser, P., & Wyeth, P. (2005). GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment*, *3*(3), 1–24.

Trefry, G. (2010). *Casual game design: Designing play for the gamer in all of Us*. Morgan Kaufmann Publishers.

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014). A Review of Games Designed to Improve Introductory Computer Programming Competencies. In *44th Annual Frontiers in Education Conference* (pp. 781–787). Madrid, Spain.

Winslow, L. E. (1996). Programming pedagogy - a psychological overview. *ACM SIGCSE Bulletin*, *28*(3), 17–22.

Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games. *Computer*, *38*(9), 25–32.