NoBug's Snack Bar: A Computational Thinking Serious Game as an Educational Platform

A. Vahldick, P. Farah, M. Marcelino, and A. Mendes

***Abstract*—This paper presents the development of an educational platform with serious gameplay to support introductory programming learning. On one hand, it presents the technical aspects of the environment and, on the other hand, based on an experimental study done, it identifies students' behavior and attitudes when playing on their own initiative. We want to verify the relations between students' performance and their engagement, satisfaction, and problem-solving strategies. The understanding of these relations may help to identify students' characteristics that may point out to the teacher the need for individualized support.**

***Index Terms*—Computer programming, Blocks-based approach, Serious games.**

I. Introdução

EXISTEM algumas iniciativas para introduzir o aprendizado de programação para a geração atual de crianças e jovens na União Europeia [1] e Estados Unidos [2]. Entretanto, ainda decorrerão alguns anos em que os estudantes dos cursos de graduação não tiveram qualquer contato prévio com essas questões introdutórias de programação na sua educação básica. Quando esses conceitos são vistos na educação básica, os estudantes podem aprendê-los num ritmo mais lento do que no ensino superior, onde se exige maior disciplina, organização e persistência no estudo [3]. A complexidade natural no processo de programação é fonte de dificuldades no aprendizado e, consequentemente, reflete-se em altos índices de desistência e reprovação logo no primeiro semestre [4]. Em estudos recentes, Portugal e Brasil estiveram entre os três países com maior índice de reprovação nesse contexto (62.1% e 55% respectivamente), mas geralmente esses índices estão entre 30% a 40% [5]. Por essa razão, a maior parte das pesquisas realizadas no ensino de programação se concentram no ensino superior [6].

Os estudantes desta nova geração, habituados com jogos e outras mídias eletrônicas, não se sentem motivados com exercícios para calcular e imprimir números na console ou numa janela, mas estão acostumados a consumirem animações, gráficos e sons e provavelmente estes são tipos de mídia que gostariam de produzir [7]. O uso de jogos vêm se integrando com o currículo para simular atividades da vida real [8] e fornecem oportunidades de aprendizado significativas na esperança de aumentar o interesse dos estudantes para o conteúdo educacional [9]. Eles podem aprender de forma personalizada (jogos conseguem se ajustar conforme o jogador) e auto assistida (os jogadores percebem os seus erros quando falham nas tarefas do jogo, que, por sua vez, instrui o jogador como realizar determinada ação) [10]. Essa é uma nova tipologia, chamada de Jogos Sérios, que são projetados para terem a instrução como objetivo primário [11], permitindo que os alunos desenvolvam novas habilidades, aprendam novos conhecimentos e fortaleçam competências existentes [12]. Os jogos conseguem atrair os estudantes, aumentando a relevância e imersão no conteúdo e auxiliam na transferência do conhecimento.

Uma boa maneira de promover o aprendizado de programação é a prática disciplinada e intensiva por parte dos estudantes [4]. Entretanto, muitas vezes eles carecem de motivação para se envolverem nessas tarefas de programação. Praticar a resolução de problemas em jogos pode ser mais motivador do que usando exercícios tradicionais, pois promove a confiança através da experiência na construção de conjuntos de padrões de soluções que serão muito úteis quando os alunos se depararem com resoluções de problemas usando linguagens de programação reais [13].

Os jogos para programar mesclam mecânicas de jogo com programação oferecendo uma experiência de programação autêntica e enriquecedora [9]. Estes jogos vêm reforçar e praticar conceitos de programação [14]. Entretanto, o uso de jogos nem sempre tem impactos positivos e é necessário considerar [15]: mudanças no currículo, nos instrumentos e nas formas com que os alunos são avaliados; que o jogo seja simples, fácil e barato de ser desenvolvido e modificado; que o jogo nem desencoraje aqueles com pouco conhecimento prévio e nem desestimule aqueles que já vem com bastante conhecimento.

Em um trabalho anterior dos autores [16] são apresentados 40 jogos desenvolvidos para apoiar o aprendizado de programação. A maioria deles não tinha elementos de jogos para promover a motivação extrínseca através da diversão. Por exemplo, não continham um sistema de pontuação ou conquistas. Os jogos se pautavam apenas no aluno cumprir as tarefas nas missões. Conquistar pontos e colecionar medalhas faz com que as pessoas se comprometam mais no jogo [17]. Ainda, a diversão permite que os alunos realizem as suas tarefas mais facilmente, mesmo que elas exijam um maior esforço [10]. Neste contexto, esse artigo apresenta um novo

A. Vahldick, Universidade do Estado de Santa Catarina, Ibirama, Santa Catarina, Brasil, adilson.vahldick@udesc.br.

P. R. Farah, Universidade do Estado de Santa Catarina, Ibirama, Santa Catarina, Brasil, paulo.farah@udesc.br.

M. J. Marcelino, Departamento de Engenharia Informática, Universidade de Coimbra, Coimbra, Portugal, zemar@dei.uc.pt.

A. J. N. Mendes, Departamento de Engenharia Informática, Universidade de Coimbra, Coimbra, Portugal, toze@dei.uc.pt.

jogo sério, chamado NoBug's Snack Bar, que incorpora alguns elementos comuns em jogos como a conquista de pontos, tabelas de classificação de jogadores, e até mesmo a personalização dos avatares dos jogadores, com a intenção de aumentar o sentido de presença, a satisfação e a motivação em vencer os desafios no jogo [18]. O jogo é uma aplicação web que contém uma área administrativa para que o professor acompanhe a evolução dos alunos. Apesar do jogo fornecer 74 exercícios, o professor consegue personalizar esses exercícios, ou inserir novos, através do carregamento de arquivos XML.

Um dos maiores problemas no aprendizado de programação se refere aos erros de sintaxe [19]. Por isso, como muitos dos jogos analisados, adotou-se uma abordagem de Programação Baseada em Blocos (PBB) porque o objetivo não é o aprendizado de uma linguagem de programação real, mas sim o desenvolvimento de habilidades e competências computacionais de resolução de problemas, também chamado de Pensamento Computacional (PC). Esta é uma abordagem para resolver problemas, projetar sistemas, e entender o comportamento humano que se baseia nos conceitos fundamentais da computação [20]. Os ambientes que seguem essa abordagem apresentam a execução de programas como animações [21] e usam notações gráficas para a produção de soluções [22]. As ações, manipulação de variáveis e estruturas de controle são representados por blocos coloridos que se encaixam seguindo a metáfora do Lego [23]. Para esses ambientes é mais importante para o aluno usar uma notação focada na lógica e construção de soluções do que se preocuparem com a gramática da linguagem [24]. Um ambiente ideal para programação introdutória deve proporcionar uma interface simples que suporte a visualização dos objetos, que tenha um editor baseado em blocos, relate as mensagens de erro de forma simples e instrua como corrigi-las, e ainda com a capacidade de executar passo-a-passo um programa [25].

Neste artigo são descritos os elementos do jogo, o seu projeto instrucional, método de avaliação das soluções submetidas pelos alunos, o ambiente de monitoramento do professor e a estrutura da definição de uma missão. Além disso, apresentam-se os resultados de uma experimentação avaliando o engajamento dos alunos (tempo e quantidade de tarefas concluídas no jogo), relacionando com o desempenho deles na disciplina e no jogo, a satisfação deles com o jogo, e confrontando essas informações com um esquema representando as estratégias dos alunos na resolução dos problemas.

II. Abordagem Construcionista no Uso de Jogos

Os computadores podem melhorar a aprendizagem quando os alunos conseguem ver os resultados concretos dos seus esforços [26]. Na abordagem construcionista de educação, os alunos coordenam a sua aprendizagem pela construção, manipulação e teste de conceitos num micromundo [27]. Um micromundo é um espaço com premissas e restrições que fornecem um contexto para que o aprendiz construa o seu conhecimento através da experimentação [26]. Os alunos aprendem pela exploração e construção nesse mundo, onde os efeitos das suas ações se refletem no que é correto ou incorreto nas suas crenças. Os resultados da aprendizagem ocorrem pela prática ativa.

Os jogos construcionistas trazem essas ideias (aprendizagem dirigida pelo aluno, construções pessoalmente significativas, ênfase em ideias significativas) em seus projetos [28]. Na última década, os ambientes e os jogos de PC materializaram a abordagem da aprendizagem construcionista. Para aprenderem conceitos abstratos de programação, os alunos necessitam construir esses conceitos por experiência prática [29]. O PC e o construcionismo compartilham o desenvolvimento de duas habilidades básicas: raciocínio procedimental e depuração. Pensar procedimentalmente envolve dividir um problema em partes menores e reconhecer padrões que podem efetivamente se repetir [26]. Depurar envolve sistematicamente tentar ajustar um pedaço de código para identificar e corrigir erros para manter o sistema executando apropriadamente [30].

Os jogos construcionistas contemplam dois princípios de projeto: as suas ferramentas e recursos devem ser expressivos e os objetivos precisam encorajar a exploração [31]. O tamanho dos blocos de construção deve permitir que o aluno expresse ideias e estratégias que são significativas no seu contexto de aprendizagem: nem tão grandes para que o jogo seja muito fácil, e nem tão pequenas para evitar o tédio ou tarefas muito difíceis. Os jogos podem recompensar por uma variedade de descobertas, não sendo limitados a uma única ou a um pequeno conjunto de estratégias vencedoras. As atividades de criação podem fazer-se de muitas formas, mas é importante que os artefatos resultantes sejam identificáveis e úteis. Além disso, o ciclo típico de interação e resposta para desenvolver o PC [32] é adequado para qualquer jogo construcionista: (1) os alunos desenvolvem, executam ou depuram a solução, (2) o jogo realiza as ações baseadas na solução submetida, e (3) o jogo fornece os resultados, respostas e suporte ao aluno. Este ciclo iterativo e interativo proporciona poderosas possibilidades para o aluno tentar e repetir as suas tentativas naquilo que acredita, e melhorar o seu conhecimento.

III. Objetivos e Metodologia de Investigação

O objetivo geral com o desenvolvimento deste novo jogo sério foi prover uma plataforma com um conjunto de tarefas de aprendizagem cobrindo os principais assuntos das disciplinas introdutórias de programação nos cursos superiores. Contudo, cada professor pode adaptar o projeto instrucional (as fases e as missões) conforme a sua necessidade. Ainda, o professor tem acesso à evolução dos alunos e o jogo destaca aqueles que precisam do seu suporte. O jogo, chamado NoBug's Snack Bar, foi projetado com três objetivos principais:

1. Usar um contexto diferente da abordagem tradicional de movimentar tartarugas e robôs, oferecendo um contexto mais significativo com base no cotidiano do público alvo, tal como uma lanchonete. Consequentemente, os comandos representam ações de mais alto nível: em vez de comandar o personagem para andar um passo de cada vez, é comandado para ir a um determinado cliente ou preparar um cachorro-quente;

2. Os alunos conseguem jogar e aprender com o mínimo de intervenção do professor. O jogo trata do desenvolvimento do estudante através de uma sequência de aprendizagem, corrigindo os seus programas enquanto ele joga;
3. Antes do aluno criar os seus próprios programas no jogo, eles experimentam boas práticas de programação. O jogo oferece soluções parciais para que os alunos completem o programa. O jogo também apresenta programas com erros para que os alunos corrijam. Além disso, o jogo apresenta soluções completas para os alunos avaliem e coloquem os comandos na sequência correta.

A pesquisa experimental conduzida para o desenvolvimento do jogo usou uma abordagem iterativa com a metodologia Design-Based Research (DBR) [33]. Através dela foi possível concentrar-se com a aplicação pragmática das teorias de aprendizagem, pois ela é projetada para fortalecer as relações entre a teoria e a prática com exemplos concretos de aprendizagem que podem ser usados e reusados no mundo real [34]. A DBR é caracterizada como um processo iterativo e intervencionista [35] que visa definir teorias que explicam o processo de aprendizagem. As primeiras iterações apresentam resultados frágeis. Entretanto, o aspecto iterativo permite que os artefatos evoluam, sejam ajustados e otimizados durante as intervenções [36]. Os investigadores testam as teorias através de protótipos e refinam as ideias até as teorias se tornarem mais robustas. Os protótipos evoluem, e consequentemente as teorias contribuem para entender as ações que induzem ou não o aprendizado [37]. Essa compreensão acontece exclusivamente através dos experimentos que produzem e são produzidas das novas teorias [36] e acontece principalmente num contexto de mundo real [33]. Alguns trabalhos descrevem o desenvolvimento de jogos sérios utilizando a DBR [38]–[40]. Eles intensivamente utilizaram prototipação (tanto no papel quanto digital), mencionam que os jogos sofrem grandes mudanças durante as intervenções, tanto na maneira de jogar, quanto na mecânica e no conteúdo de aprendizagem.

A investigação tratada nesse trabalho foi conduzida por quatro ciclos de interações. O primeiro ciclo, como uma experiência piloto, foi realizado em sala supervisionado em dois momentos e locais distintos: o primeiro na Universidade de Coimbra (UC) por um investigador e o segundo na Universidade do Estado de Santa Catarina (Udesc) por um professor. O objetivo desse primeiro ciclo foi de perceber as oportunidades de aprendizagem e diversão com o jogo. Os resultados dessa avaliação estão em [41].

Os três ciclos seguintes foram conduzidos em disciplinas de primeiro semestre por aproximadamente dois meses cada, para que fosse possível deduzir pequenas generalizações e, durante o próprio ciclo, fossem feitas as mudanças para que a aprendizagem e diversão pudessem ser aperfeiçoadas com os próprios alunos. O segundo ciclo envolveu 60 alunos na UC [42]. Na Udesc foram conduzidos o terceiro ciclo com 36 alunos [43] e o quarto com 33 alunos.

Nesses três ciclos, os alunos tinham a liberdade de acessar onde, quanto e quando quisessem. Quando alcançassem determinado ponto de evolução no jogo, eles tinham que responder um questionário de avaliação (EGameFlow) com intuito de medir a satisfação deles com o jogo. EGameFlow [44] é um instrumento adaptado do GameFlow [45] para avaliar jogos sérios. O jogo é mensurado em oito dimensões baseadas no fluxo de [46]: concentração (a quantidade de informação ou tarefas extras e desnecessárias nas atividades de aprendizagem?), clareza (os objetivos são compreensíveis e explícitos durante todo o jogo?), suporte (o aluno recebe suporte suficiente e no momento correto?), desafios (os desafios são coerentes com as habilidades e o conhecimento do aluno?), autonomia (o aluno tem o sentimento de que comanda o jogo ao conhecer os controles necessários para realizar todas as ações?), imersão (o aluno sente-se envolvido profundamente nas tarefas do jogo?), interação social (existem recursos no jogo para promover a competição ou cooperação entre os alunos?), e a melhoria no conhecimento (o aluno consegue aumentar o seu nível de conhecimento e as suas habilidades enquanto cumpre os objetivos do jogo?). O EGameFlow contém 42 questões com níveis de concordância numa escala Likert de sete níveis. O valor final classifica o senso geral de diversão no jogo em uma escala de 0 a 100.

Além das respostas a estes questionários, também foram usados os *logs* de acesso e as soluções das tarefas como fonte para avaliação dos ciclos. A seção IV apresenta a última versão do jogo, experimentada no quarto ciclo, e a seção V os resultados dessa avaliação.

IV. NoBug's Snack Bar

A. O Ambiente do Jogo com seus Elementos Instrucionais e de Diversão

NoBug's Snack Bar é um jogo web inspirado no gênero de gestão de tempo. O jogador controla o atendente de uma lanchonete. O jogo é baseado no cumprimento de tarefas em missões que podem ser resolvidas em até 5 minutos. No jogo, os clientes pedem uma combinação de comidas e bebidas, e o atendente (controlado pelo jogador) precisa ir aos locais corretos para preparar os alimentos, e servir aos clientes. A missão encerra quando o jogador atende todos os pedidos.

A Fig. 1 ilustra um exemplo de missão, à esquerda está a configuração da lanchonete e o pedido do cliente e à direita está uma possível solução. O primeiro bloco indica que o atendente vai até a posição 2 do balcão (representado por A). Então ele guarda o pedido do cliente em uma variável chamada "*pedido*". O terceiro bloco representa um condicional e compara o valor da variável "*pedido*" com uma constante chamada *softDrink*. Se eles tiverem o mesmo valor então o atendente vai até a geladeira (B), pega a bebida de acordo com a variável e armazena essa bebida na variável "*bebida*".

Caso contrário, ele vai até a caixa de frutas (C), pega as frutas de acordo com a variável "*pedido*" e armazena na variável "*frutas*". Em seguida, o atendente vai à máquina de sucos (D), prepara o suco e armazena o resultado na variável "*bebida*". Após o bloco condicional, o atendente retorna ao cliente e entrega o "*pedido*". Se a entrega combina com o que o cliente pediu, o jogador atende os objetivos da missão. Do contrário, o cliente fica irritado, o jogador recebe uma mensagem de erro, a execução é interrompida, mas o aluno consegue alterar a sua solução e tentar novamente.

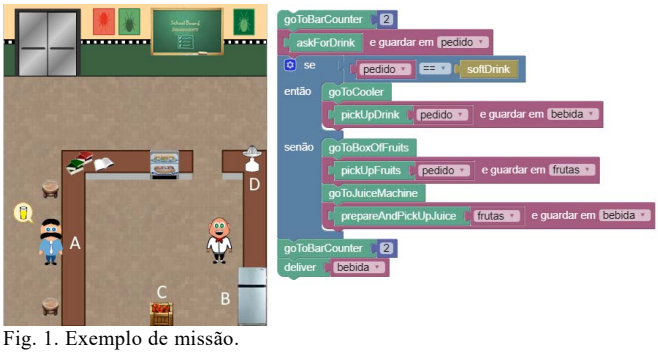

Fig. 1. Exemplo de missão.

O *frontend* do jogo foi codificado em HTML5, usando o framework Blockly [47]. Do lado servidor foi desenvolvido em Java. A Fig. 2 apresenta a interface do jogo onde o jogador resolve os problemas. O jogador cria a sua solução da missão na área central. A área de animação (à esquerda) permite a observação das vontades dos clientes e dos movimentos do atendente baseado na solução em blocos. O jogador consegue executar ou depurar a sua solução. Quando ele depura, o jogo apresenta à direita a lista de variáveis e os seus valores e executa um bloco a cada clique do botão de depuração. Esses botões para o controle da execução estão exatamente abaixo da área de animação: executar, depurar e interromper a execução/depuração, o controle de velocidade da execução da animação (a tartaruga para o mais lento até à lebre para o mais veloz) e a apresentação da quantidade de testes (dentro do círculo preto) a serem executados na solução com a quantidade de testes bem-sucedida (dentro do retângulo vermelho). Esses testes unitários existem para que uma solução seja confrontada com diferentes entradas (vontades dos clientes) e garantir que a solução do aluno não tenha sido desenvolvida para atender somente uma configuração de vontade dos clientes. No exemplo da figura, a solução será submetida a 10 diferentes configurações de clientes e uma delas já foi bem-sucedida. Abaixo desses botões encontra-se uma janela com todos os objetivos da missão e quais já foram cumpridos. Na próxima subseção será detalhada a definição desses objetivos e a verificação do seu cumprimento.

Em relação aos comandos disponíveis para produção da solução, existem aqueles que não tem uma saída esperada (p.e. *goToBarCounter*, *deliver* e *talk*) e outros representam funções (p.e. *askForFood, askWantHowManyDrinks* e *pickUpDrink*). Quanto aos nomes dos comandos, o Blockly permite que eles possam ser facilmente alterados. Foi decidido usar primeiramente em inglês – pois essa é a realidade quando forem programar com uma linguagem real – e verificado se os alunos tinham algum tipo de reclamação em relação ao idioma. Como durante o experimento não houve queixas sobre isso, foram mantidos os blocos em inglês.

Os pontos ganhos são contados pelas tentativas: os alunos ganham mais pontos quanto menos tentativas fizerem para cumprir os objetivos da missão. Na Fig. 2 é possível observar como funciona essa sistemática onde existem três estrelas (com *x30* ao lado) e três setas. Isso representa que, nessa missão, cada estrela tem três tentativas. A Fig. 2 exemplifica que o aluno está na quinta tentativa porque uma das estrelas e uma das setas está incolor. Cada estrela está relacionada com uma quantidade de pontos (30 pontos na Fig. 2). Após perder as três estrelas, o aluno ainda ganha pontos, mas em menor valor (por exemplo, 5 pontos) quando cumpre todos os objetivos da missão. Cada missão tem a sua própria quantidade de tentativas por estrela, assim como a quantidade de pontos por estrela.

Existe uma outra forma de ganhar pontos, moedas, ou mesmo pontos extras nas avaliações (provas, trabalhos, etc.) da disciplina através de um Sistema de Conquistas. O professor pode definir regras em como os alunos podem obter essas recompensas (por exemplo, finalizar todas as missões de um nível até uma determinada data) e o que eles recebem em troca (por exemplo, pontos e moedas no jogo, ou pontos extras no diário de classe).

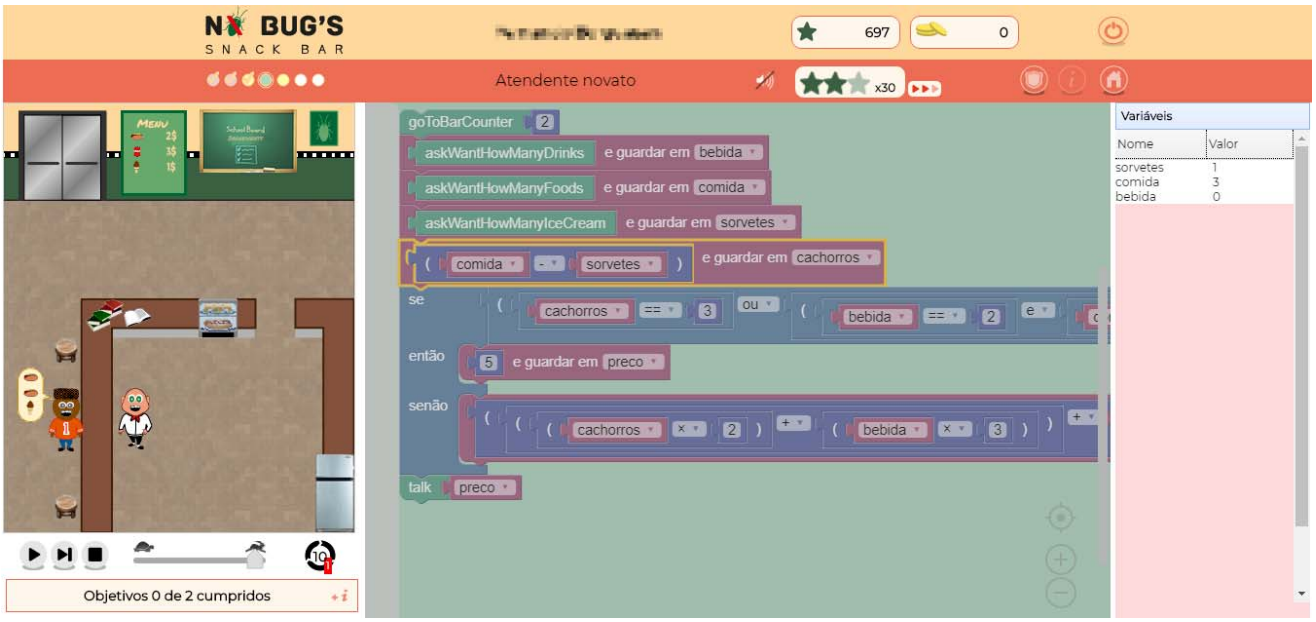

Fig. 2. Interface do jogo NoBug's SnackBar.

Existem três formas de classificar os alunos e exibi-lo numa tabela de classificação: pela quantidade de pontos ganhos, pela quantidade de tempo consumido para vencer as missões e pela quantidade de tentativas para vencê-las. As tabelas apresentam a posição, o nome e os valores dos 10 melhores jogadores e do jogador atual, caso ele não esteja nessa classificação. Isso evita o constrangimento em relação aos seus colegas. Um aluno pode estar em posições diferentes em cada uma das classificações. A intenção é que isso estimule o aluno em melhorar algum aspecto, como resolver em menor tempo ou menos tentativas uma missão.

Os jogadores conseguem configurar a aparência de seu atendente. No início a personagem inicia careca e existem poucas opções como configurar a cor dos olhos e da pele. À medida que os alunos vão conquistando pontos, novas opções ficam disponíveis para configurar: chapéu do *chef* e seu dólmã, cores especiais de pele, cortes e cores de cabelo para as mulheres, ou cortes e cores de barba e bigode para os homens. A Fig. 3 exemplifica a janela onde são configuradas as cores dos olhos e da pele, assim como o corte e cor do bigode do avatar.

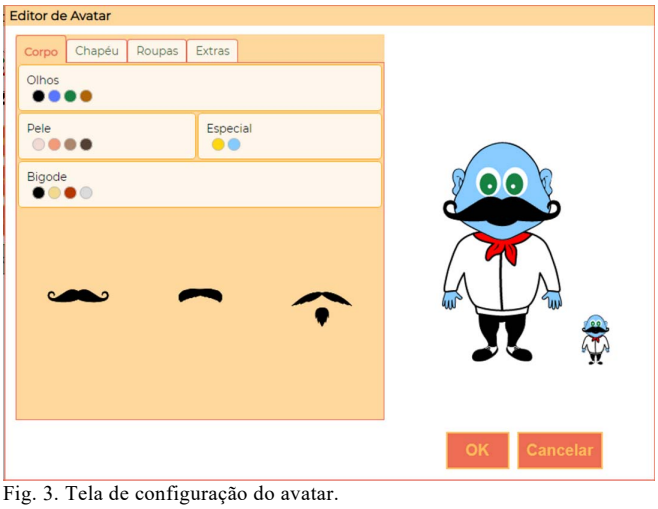

Fig. 3. Tela de configuração do avatar.

A sequência instrucional do jogo é ilustrada na Fig. 4. Cada círculo representa uma fase. As setas representam os requisitos entre as fases e entre parêntesis é indicada a quantidade de missões em cada fase. As 10 fases cobrem os conceitos iniciais comuns em disciplinas introdutórias de programação. Essa sequência foi inspirada em [48] que apresentou um grafo com os conceitos-chave para o aprendizado de algoritmos. Nele, os autores destacaram como conceitos fundamentais o endereçamento, memória, dados, sequência e instrução. Os autores também definem os conceitos transformativos (variável, atribuição, expressões, condicionais, laços de repetição e funções) como aqueles que agregam os conceitos fundamentais adicionando a eles novos significados. As 74 missões estão organizadas por conteúdo, da seguinte forma: fases 2 a 4 (19 missões) são sobre manipulação de variáveis, fases 5 a 7 (22 missões) sobre condicionais e fases 8 a 10 (24 missões) sobre laços de repetição. A fase 1 apresenta as ferramentas do jogo. As fases em branco incluem missões em que o estudante aprende os conceitos básicos e elementares sobre o conteúdo. Inclusive elas são fases obrigatórias. As fases em cinza-claro permitem ao estudante praticar novas e mais complexas situações para se aperfeiçoar nos conceitos dessas fases. As fases em cinza-escuro possuem missões que desafiam o estudante para que ele possa se aprofundar nos conceitos abordados.

A sequência dos tipos de tarefas a serem feitas dentro das missões (Tabela I) foi inspirada nas categorias do domínio cognitivo proposto pela Taxonomia Revisada de Bloom [49]. Nas fases introdutórias, as missões iniciais introduzem novos conceitos apresentando soluções e uma questão a ser respondida (tarefas de múltipla escolha) e/ou apresenta soluções incorretas que devem ser corrigidas (tarefas de correção de erros). À medida que o aluno avança numa fase, os tipos mudam para tarefas de ordem mais alta, como organizar blocos e completar nos espaços. As últimas missões solicitam que o aluno construa a solução a partir do zero. Nas fases de aperfeiçoamento, primeiro uma missão é usada com um dos primeiros quatro tipos de tarefas para reintroduzir um conceito ou apresentar um novo equipamento para a lanchonete. As missões seguintes sempre serão do tipo criação de soluções. Nas fases de aprofundamento todas as missões são de criação, e muitas delas possuem algumas restrições na solução, por exemplo, na quantidade de blocos ou variáveis a serem usadas.

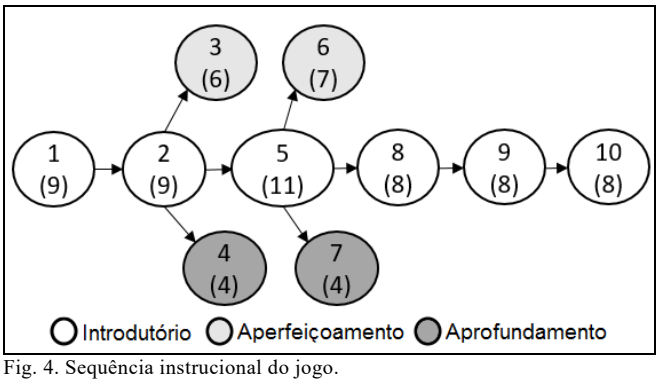

Fig. 4. Sequência instrucional do jogo.

TABELA I
TAREFAS DAS MISSÕES

Tipo da Tarefa	Descrição da Tarefa
Múltipla escolha	O jogo fornece uma solução e uma questão sobre ela com quatro opções de resposta. O aluno seleciona uma das opções. O jogo executa a solução e verifica a resposta do aluno.
Corrigir erros	O jogo fornece uma solução com erros. O aluno precisa alterar a variável que está sendo referenciada ou trocar operadores de comparações ou lógicos.
Organizar blocos	Todos os blocos já estão disponíveis espalhados na área de trabalho e o aluno precisa organizá-los na sequência correta.
Completar nos espaços	O jogo fornece uma solução parcial faltando alguns blocos. O aluno completa a solução adicionando os blocos faltantes.
Criação	O aluno cria sua solução a partir do zero.

Existe um sistema de dicas que apresenta uma mensagem ao aluno caso atenda alguma condição de disparo. Essas dicas estão divididas em dois momentos. Um dos momentos acontece durante a produção da solução que verifica constantemente o que o aluno está a produzir e a mensagem é

disparada quando o aluno fica inativo por algum tempo. O outro momento acontece quando ocorre algum erro na execução ou a execução finalizou sem conquistar todos os seus objetivos, por exemplo, sugerindo que o aluno verifique se uma variável utilizada como parâmetro foi previamente inicializada, como é o caso de tentar entregar um pedido ao cliente, e o atendente realmente pegou o produto da prateleira.

Nas missões do tipo criação existem a opção para o aluno invocar um assistente pelo custo de todas as estrelas. Esse assistente mostra a solução da missão em pseudocódigo (Fig. 5) que permite ao aluno avançar ao montar a solução interpretando o pseudocódigo fornecido.

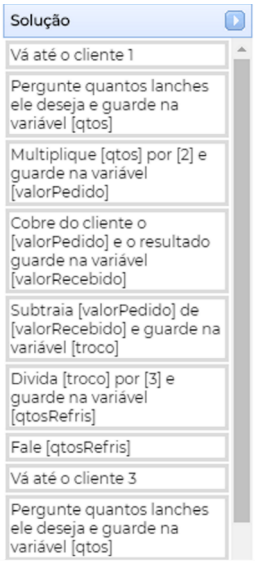

Fig. 5. Assistente de pseudocódigo.

B. Definição das Missões e Avaliação das Soluções Submetidas

A seção anterior apresentou uma instância do jogo, ou seja, uma proposta experimentada de estrutura de fases e missões. Os resultados da experimentação serão apresentados em seções subsequentes. Para permitir que o jogo seja um sistema aberto (=plataforma), todas essas definições são feitas em um banco de dados acessado pelo jogo. As missões são definidas em um arquivo XML. Como um trabalho futuro, existe a necessidade de desenvolver um editor de fases e missões. Por enquanto, o professor precisa desenvolver as missões conhecendo as definições dos elementos XML. A título de ilustração, a Tabela II apresenta os elementos desse documento. O elemento <*objectives*> será o foco desta seção para elucidar o funcionamento no jogo do processo de definir as tarefas a serem cumpridas pelo estudante e a verificação do jogo com o cumprimento dessas tarefas.

A Fig. 6 ilustra um exemplo do elemento que define os objetivos (<*objectives*>). Nele constam atributos do tipo de missão (*missionType*, no caso *fillInGaps* indica o tipo "Completar nos espaços") de acordo com a Tabela I, em relação à pontuação (*xpIndividual*, *xpFinal* e *xpTotalRun*), e limites na quantidade de comandos (*commQtd*) e pontuação extra (*maxCommands* e *maxCommandsRewardCoins*). Porém, ainda existem outros 6 atributos, não demonstrados aqui, para habilitar botões e definir outras restrições.

TABELA II
ELEMENTOS DAS MISSÕES NO XML

Elemento XML	Descrição do Elemento
<*explanation*>	Conteúdo de aprendizagem e a descrição da tarefa
<*hints*>	Regras que definem quando uma dica é mostrada
<*help*>	Assistente de pseudocódigo
<*commands*>	Comandos disponíveis para criar a solução
<*customers*>	Configuração dos clientes e seus desejos
<*objectives*>	Objetivos, restrições do jogo e pontuação
<*xml*>	Blocos iniciais sugeridos na solução

Cada elemento <*objective*> define um objetivo na missão. No exemplo, a missão contém 3 objetivos a serem cumpridos. O conteúdo desse elemento define o tipo do objetivo. No exemplo, existe o objetivo (1) *deliver* em que ele é cumprido quando tiver atendida a vontade completa do cliente na posição 2 do balcão (definido pelos atributos *pos* e *place*); (2) *askWantHowManyFoods* em que ele é cumprido se foi usado esse tipo de bloco na solução para o cliente na posição 2 do balcão; e (3) *callTimes* que é cumprido se o bloco "*para*" foi usado uma única vez na solução. Alguns objetivos são usados para guiar o aluno na construção da solução, como é o caso do segundo objetivo em que ele precisa usar o bloco que pergunta ao cliente quanta comida ele deseja. Existem outros 19 tipos de objetivos.

```
<objectives xpIndividual="50" xpFinal="10"
 xpTotalRun="6" commQtd="15"
 maxCommands="11" missionType="fillInGap"
 maxCommandsRewardCoins="2" >
    <objective pos="2" place="counter">
      deliver
    </objective>
    <objective pos="2" place="counter">
       askWantHowManyFoods
    </objective>
    <objective block="para" times="1"
          type="controls_for">
      callTimes
    </objective>
</objectives>
```
Fig. 6. Exemplo da definição de objetivos.

O processo de verificar a solução do aluno não é baseado em analisar o código produzido, mas se durante a execução da missão foram cumpridos os objetivos estabelecidos na missão. Alguns objetivos são verificados durante a execução e outros após. Seguindo o exemplo da Fig. 6, durante a execução o jogo verifica se foi entregue o pedido ao cliente e se foi usado o bloco *askWantHowManyFoods*, e após a execução se foi usado no máximo 1 vez o bloco "*para*". Essa abordagem permite que o aluno construa a sua solução da forma que ele conseguiu compreender o problema, e não a comparação da estrutura esperada da solução.

C. Ambiente de Monitoramento

O processo ensino-aprendizagem envolve pelo menos dois atores: o aluno e o professor. Por isso, é importante considerar

as duas visões quando se desenvolve qualquer aparato tecnológico para melhorar esse processo. O professor precisa ter um meio de acompanhar a evolução individual de cada aluno e intervir naqueles que precisam de seu suporte. A Fig. 7 ilustra a página disponível para o professor como um mapa que mostra todos os alunos com as conclusões de suas missões. Cada coluna representa uma missão e cada linha um aluno. As cores nas células mostram o grupo a que a missão do aluno pertence com base na quantidade de tentativas da turma. As células em verde são as missões concluídas com base na quantidade de tentativas dentro da amplitude interquartil da turma. Em vermelho são as missões concluídas com a quantidade de tentativas além do limite superior da amplitude interquartil (discrepância máxima). Em branco representam-se missões não concluídas, ainda que o aluno tenha já feito algumas tentativas. Em amarelo são indicadas as missões não concluídas, mas que apresentam já uma quantidade discrepante de tentativas. Os números representam a quantidade de tentativas. O professor pode clicar nesses números para visualizar cada uma das tentativas dos alunos. A tentativa do aluno engloba o código produzido, os objetivos alcançados e o erro fornecido pelo jogo que tenha interrompida a execução ou pelo não atendimento de todos os objetivos. Ele pode, por exemplo, auxiliar aqueles alunos que ainda não concluíram a missão, mas já tem uma quantidade discrepante (células em amarelo) que pode representar alguma dificuldade do aluno em solucionar o problema. Esse mapa pode ser ordenado pelo nome dos alunos, pela quantidade de tentativas discrepantes (os alunos com maior discrepância ficam em primeiro), quantidade de missões feitas (os alunos com menos missões concluídas ficam em primeiro) e tempo consumido para concluir a missão (os alunos com maior tempo ficam em primeiro). Conforme as ordenações escolhidas também vão mudando os valores apresentados na célula. Por exemplo, se foi selecionada a ordenação por tempo consumido, as células apresentarão em minutos esse tempo. A ordenação privilegia sempre deixar por primeiro os alunos que supostamente precisam de mais atenção do professor.

V. Resultados

Conforme descrito na seção III. Metodologia de Investigação, foram conduzidos quatro ciclos de experimentação. Essa seção apresenta alguns resultados do quarto ciclo, com 33 alunos (87,9% homens e 12,1% mulheres) matriculados na disciplina de Introdução à Programação do Bacharelado em Engenharia de Software da Udesc. A idade média deles foi de 22,5 anos (±4,66). Em relação aos hábitos de jogar, 33,3% declarou jogar diariamente. No outro extremo, 24,2% declarou jogar raramente. Quanto ao conhecimento prévio em programação, 81,8% deles declararam não ter qualquer experiência ou contato prévio.

O jogo foi experimentado diferentemente nos três últimos ciclos (do total de 4 ciclos). No segundo ciclo o jogo foi fomentado pelo professor para ser usado como uma ferramenta adicional para prática dos alunos que apresentavam dificuldades nas aulas. Não havia explanações do jogo por parte do professor. No terceiro ciclo o jogo foi usado integrado às aulas, com o professor explicando as estruturas da linguagem nas aulas teóricas e os alunos jogavam nas aulas práticas e extraclasse. No quarto e último ciclo, o professor apresentou o jogo na primeira aula, e os alunos tiveram 3 semanas para usar o jogo sem aulas presenciais ou qualquer suporte pelo professor. A única assistência era através do monitor da disciplina e de um dos investigadores acompanhando a distância e intervindo por e-mail. Ao final da terceira semana, foi aplicada uma prova com questões que deveriam ser respondidas por blocos. As questões foram comuns a qualquer exame de introdução à programação, como cálculo de fatorial (sem recursividade), calcular IMC (Índice de Massa Corporal) e divisão de números usando sucessivas subtrações.

Nessa seção será analisado o engajamento dos alunos (a relação entre o desempenho acadêmico e a experiência no jogo, mais o uso do assistente de pseudocódigo), a aprendizagem percebida, a satisfação do aluno (através do acesso aos elementos do jogo para motivação extrínseca com

Fig. 7. Monitoramento das tarefas dos alunos.

o sistema de conquistas, classificação dos jogadores e configuração do avatar) e as estratégias de resolução adotadas pelos alunos ao tentarem resolver os problemas no jogo.

A. Relação do Jogo com o Desempenho Acadêmico

O desempenho acadêmico foi aferido através das notas de duas provas. A primeira prova envolveu questões para responderem com blocos e a segunda prova em Java. A primeira prova foi feita por 30 alunos e a média da turma foi 5,4 (±2,33) onde 33% deles (n=10) obteve nota maior ou igual a 7,0 (numa escala de 0 a 10). A segunda prova foi feita por 25 alunos e a média da turma foi 6,0 (±2,48) onde 48% deles (n=12) obteve nota maior ou igual a 7,0. Para analisar o impacto da experiência no jogo com o desempenho foi examinada a correlação bivariada de Pearson entre as notas dos dois exames, a quantidade de missões concluídas e o tempo despendido para resolver as missões. A Tabela III mostra o resultado do teste. Foram encontradas correlações moderadas positivas entre a quantidade de missões concluídas e as notas dos exames, assim como correlações fracas entre o tempo e as notas dos exames. Com isso, pode-se constatar que quanto mais os alunos jogam, maior é a probabilidade de terem um desempenho melhor nos exames. O tempo gasto não tem tanta relação com o desempenho.

Nas análises seguintes, a turma foi dividida em dois grupos usando a nota da primeira prova com o valor 7,0 como corte (G1-com notas abaixo de 7,0 e o G2-com notas acima ou iguais a 7,0). Além disso, para dar maior significância à análise, considerou-se somente os alunos que tenham concluído pelo menos 50% das missões do tipo de tarefa "Criação" (n=33), pois o objetivo principal no ensino de programação é o aluno desenvolver as suas soluções para os problemas que lhe são propostos. Como resultado, o Grupo 1 incluiu 8 alunos e o Grupo 2 incluiu 9 alunos.

TABELA III
CORRELAÇÃO ENTRE AS PROVAS E A EXPERIÊNCIA NO JOGO

	Prova 1	Prova 2	Total de missões	Tempo despendido
Prova 1	-	0,770**	0,666**	0,452*
Prova 2		-	0,641**	0,410*
Total de missões			-	0,767*
Tempo despendido				-

** p<0,01 * p<0,05

B. Aprendizagem Percebida

A aprendizagem percebida é um conjunto de crenças e sentimentos em relação a aprendizagem ocorrida e reflete o senso do estudante que algum novo conhecimento foi adquirido e alguma nova compreensão foi alcançada, mesmo que esse conhecimento e compreensão subjetivos esteja em contraste com o desempenho acadêmico [50]. A aprendizagem percebida representa o grau de confiança que o aluno tem em relação ao seu domínio de um dado conhecimento.

Para medir o sentimento dos alunos sobre a sua aprendizagem, foi aplicado um questionário (Tabela IV) com cinco níveis de concordância na escala Likert (1-discordo totalmente e 5-concordo totalmente) antes de realizarem a primeira prova. Com isso é possível avaliar a relação entre a aprendizagem medida pela prova e a aprendizagem percebida. A Fig. 8 apresenta os resultados desse questionário, onde a primeira coluna de cada questão é relativa ao G1 e a segunda ao G2. Visualmente as médias do G2 são maiores que do G1. Através de testes não-paramétricos Mann-Whitney verificou-se em quais itens existem diferenças significativas entre os dois grupos. Em três itens foram encontradas diferenças significativas: o G2 sentiu maior confiança quanto à aprendizagem de laços de repetição (Q4), observou uma maior relação das missões no jogo com o conteúdo das aulas (Q9) e tem mais certeza na recomendação do uso do jogo para o semestre seguinte (Q10). Nos demais itens não existiram diferenças significativas. Porém, ao totalizar os itens, pode-se concluir que os alunos do G2 tiveram mais confiança na sua aprendizagem do que os do G1 (p=0,000).

C. Diversão Percebida

Para identificar o nível de satisfação dos alunos com a experiência do jogo, eles responderam 26 questões do EGameFlow. Foram executados testes não-paramétricos Mann-Whitney para avaliar em quais itens existiam diferenças significativas entre os grupos. Apenas um item apresentou diferenças com p=0,011: "As metas globais do jogo foram apresentadas com clareza". Analisando o Mean Rank foi possível concluir que o G2 concorda mais que o G1 com essa questão. Também foi executado o teste com o total dos itens, e não houve diferenças significativas. Com isso, pode-se concluir que independente do desempenho acadêmico, os alunos tiveram a mesma percepção quanto à sua diversão, a saber, a pontuação média do EGameFlow para o G1 foi 3,55 e para o G2 foi 3,64 numa escala de 1 a 5. Essa escala serve de comparação entre jogos ou grupos de jogadores no mesmo jogo. Mas, se for de interesse atribuir uma nota para o NoBug's seria o G1=63,75 e G2=66.

TABELA IV
QUESTIONÁRIO PARA AFERIR A APRENDIZAGEM PERCEBIDA

#	Descrição da Questão
Q1	Eu aprendi com o jogo a raciocinar logicamente (sequência de passos)
Q2	Eu aprendi com o jogo como manipular variáveis
Q3	Eu aprendi com o jogo a usar condicionais
Q4	Eu aprendi com o jogo a usar laços de repetição
Q5	Eu aprendi com o jogo que facilita o meu trabalho dividir o problema em partes menores
Q6	Eu aprendi com o jogo como é importante depurar para resolver os erros
Q7	Eu aprendi coisas novas com o jogo que foram úteis na disciplina
Q8	O jogo fez-me compreender melhor alguns assuntos que eu já tinha visto
Q9	Existe grande relação entre as missões no jogo e o conteúdo das aulas
Q10	Eu recomendo usar o jogo para essa disciplina no próximo semestre.

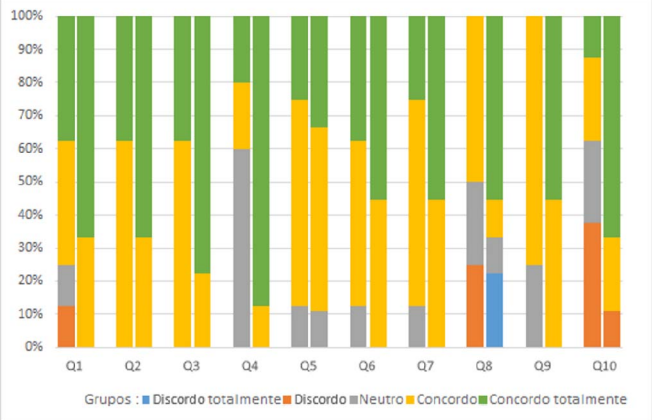

Fig. 8. Aprendizagem percebida entre os grupos.

D. Diversão Medida

Existem três elementos no jogo que não estão relacionados diretamente com a aprendizagem, mas com recursos para manter a motivação em continuar jogando para ganhar pontos: sistema de conquistas, classificação dos jogadores e a configuração do avatar. A diversão medida é considerada a assiduidade em acessar cada um desses elementos. Deseja-se identificar se existe alguma preferência entre os três itens com os dois grupos para futuramente poder explorar melhor a motivação e o aprendizado.

A medição desses itens aconteceu quando o aluno clica para abrir a janela de personalização do avatar ou a janela conquista de recompensas, ou quando manipula as tabelas de classificação. Analisando o *log*, identificou-se em quantas sessões (a cada autenticação) cada um desses recursos foi acessado pelos alunos. A Tabela V apresenta a quantidade de vezes que os alunos de cada grupo fizeram a autenticação, acessaram as janelas do avatar, da tabela de classificação e das recompensas.

TABELA V
ANÁLISE DA DIVERSÃO MEDIDA

Grupos	Número de Autenticações	Avatar	Classificação	Recompensas
1	383	52 (13,6%)	61 (15,9%)	26 (6,8%)
2	475	48 (10,1%)	130 (27,4%)	56 (11,8%)
p-value	-	0,115	0,000*	0,013*

*p-value<=α=0.05

Para comparar se existem diferenças significativas entre os grupos, foram executados testes de quiquadrado (Chi-square) para verificar se as proporções são as mesmas entre os dois grupos. A proporção de cada item foi calculada em relação à quantidade de autenticações. O recurso mais usado pelo G2 foi a visualização da tabela de classificação, seguido pela consulta das conquistas e personalização do avatar. Já no G1, a tabela de classificação também foi o mais usado, seguido pela personalização do avatar e recompensas. Ambos os grupos tiveram a consulta às tabelas de classificação como o mais usado. Entretanto, o G2 teve uma maior incidência de uso que o G1. Isso pode demonstrar a motivação intrínseca dos alunos do G2 em se manterem em melhores posições, e para isso precisam ter melhor desempenho no jogo.

E. Uso do Assistente de Pseudocódigo

Analisar o quanto o assistente foi usado permite identificar a quantidade de alunos que considerou que ele poderia ser de utilidade. Além disso, a diferença entre o momento em que o assistente foi usado e o momento da conclusão da missão pode fornecer indícios sobre o conhecimento do aluno naquele momento, pois, se mesmo com essa ajuda ele demora a concluir a missão, provavelmente ele ainda não tem assimilado bem o conhecimento sobre o assunto em causa. A Fig. 9 apresenta a comparação do uso do assistente entre os dois grupos. Ainda foi adicionado outro grupo referente aos alunos que não foram classificados nos dois grupos anteriores (aqueles que fizeram menos que 50% das missões do tipo "Criação"). Em algumas missões o assistente foi usado com exclusividade por alunos do G1 (por exemplo, 37 e 38) ou por alunos do G2 (16, 32 e 55). Um teste não-paramétrico Mann-Whitney foi executado para comparar a quantidade de vezes entre os dois grupos resultando em p=0,001 e, uma vez que o Mean Rank do G1 (alunos com notas mais baixas) foi superior, pode-se concluir estatisticamente que esse grupo usou mais o assistente. Para verificar o quão eficaz foi o uso do assistente, calculou-se a diferença entre a tentativa em que o assistente foi disparado e a tentativa de conclusão da missão. O assistente foi usado 82 vezes por 18 alunos (entre os três grupos). Em média foram necessárias 4,9 (±7,0) tentativas entre o momento de usar o assistente e de concluir a missão, com uma mediana de 2 tentativas.

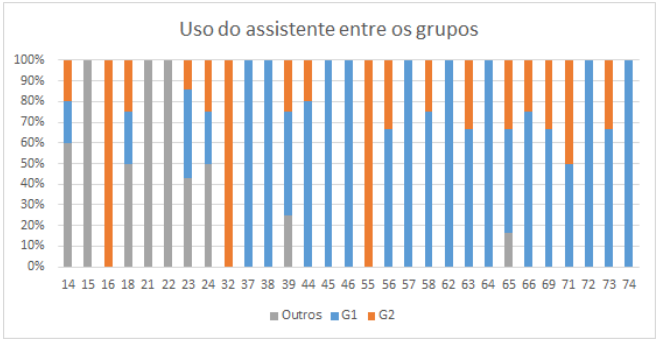

Fig. 9. Comparação do uso do assistente entre os grupos.

F. Comportamento quanto à Resolução de Problemas

As subseções anteriores apresentaram análises estatísticas comparando as percepções, preferências e usos dos recursos do jogo entre dois grupos criados com base no desempenho acadêmico. Nessa subseção será analisada a forma como os alunos usaram o jogo para resolver os problemas. Para isso, serão consideradas cinco tipos de ações que eles podiam fazer no jogo:

1-Explicação (EXP): é o momento em que o aluno acessa a explicação da missão;

2-Depuração (DEB): é o momento em que o aluno decide executar passo-a-passo a sua solução;

3-Execução (RUN): é o momento em que o aluno decide executar a sua solução;

4-Desenvolvimento (DEV): é o momento em que o aluno modifica a sua solução, adicionando, removendo ou alterando blocos;

5-Revisão (REV): é o momento em que o aluno sai de uma missão que não concluiu, e voltou a experimentar outras missões já concluídas, executando ou depurando, e por fim voltando à missão original.

Foi feita a análise do *log* e contadas as vezes em que aconteceu a transição entre uma ação e outra, distinguindo entre os alunos do G1 e G2. Foram executados testes de quiquadrado (Chi-square) para verificar as proporções entre os dois grupos. Para cada transição em que as proporções não forem iguais, é identificado como um comportamento particular daquele grupo. Com base nas transições que apresentaram diferenças significativas, foi desenvolvido um grafo para cada grupo procurando identificar os comportamentos mais evidentes em cada um deles (Fig. 10 e 11).

Ao observar o grafo do G1, fica evidente que após uma execução (run) fracassada, esses alunos logo em seguida tendem a modificar (dev) a solução e depois usam os recursos de depuração (deb). Isso pode ser entendido que a partir da execução eles já confiam que sabem o que é necessário alterar na solução. Depois da alteração, então preferem usar os recursos de depuração. Em relação ao G2, após reler a explicação (exp) eles costumam executá-la (run) provavelmente para tentar identificar o erro não percebido mesmo após a leitura. Após a execução (run) fracassada, esse grupo prefere usar mais a depuração (deb), indicando um cuidado maior em identificar o erro, em vez de ir logo alterar a solução. Enquanto estão desenvolvendo (dev) a solução, esses alunos ainda utilizam soluções anteriores (rev). É interessante verificar a diferença entre o G1 e G2 após modificar a solução: o G1 prefere mais utilizar a depuração, e o G2 prefere mais a execução. Como um trabalho futuro, o ambiente de monitoramento pode incorporar a identificação desses padrões para que o professor possa intervir sugerindo ao aluno adotar outro comportamento.

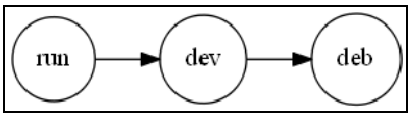

Fig. 10. Comportamento evidente do G1.

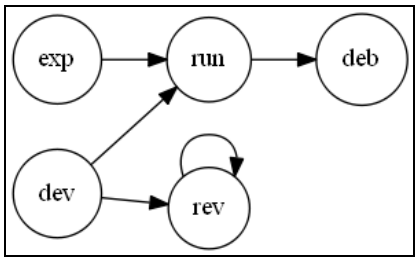

Fig. 11. Comportamento evidente do G2.

VI. Conclusões

Este trabalho apresentou um novo jogo inspirado no fato de não ter sido encontrado outro jogo que permitisse a customização do projeto instrucional, tanto na definição das tarefas de aprendizagem, quanto na organização dessas tarefas. Além disso, os jogos não apresentam uma visão das interações dos alunos para que os professores possam monitorá-los durante o jogo, e assim auxiliá-los no suporte em sala de aula. Ao oferecê-lo como um ambiente de aprendizagem, extrapolou-se o limite de ser apenas um jogo, para ser uma ferramenta de apoio em que o professor possa auxiliar os alunos a vencerem as barreiras iniciais na aprendizagem de programação.

Os ciclos de investigação não limitaram os alunos nem em tempo e nem em espaço, pois foi conduzida para livre uso em quaisquer horários e locais, permitindo avaliar resultados mais próximos da realidade em termos de diversão e aprendizagem. Nesse artigo foram apresentados resultados do último ciclo de experimentação. Foram encontradas correlações moderadas positivas entre o desempenho nas notas das provas dos alunos com a quantidade de missões concluídas. Os alunos com notas para aprovação na disciplina (>=7,0) se sentiram mais confiantes no aprendizado com o jogo, principalmente com o assunto de laços de repetição. A diversão percebida foi a mesma independente do desempenho na disciplina. Os alunos com desempenho inferior (<7,0) usaram mais o assistente de pseudocódigo. Depois de acionado o assistente, os alunos concluíram as missões em média com 4,9 tentativas.

Foram usadas algumas formas para chamar a atenção inicial aos alunos para o jogo, como conquistar pontos para personalizar o avatar, três tabelas de classificação e um sistema de conquistas. Porém, um elemento de jogo que foi testado e que não é frequentemente considerado nas publicações, é a própria conquista de pontos. Ao constatar que os alunos gostavam de acessar a tabela de classificação, pode-se concluir, que mesmo não sendo a pontuação um objetivo primário de um jogo sério, a sua utilização aumenta a atração do jogo.

Foi possível identificar uma diferença no comportamento em como resolviam as missões entre os alunos que tiveram desempenho acadêmico distinto, o que pode indicar o mesmo comportamento quando resolvem exercícios de programação: os alunos melhores classificados acessam soluções anteriores para auxiliá-los a resolverem uma nova missão, enquanto os outros, costumam realizar um ciclo de tentativa e erro.

Agradecimentos

A.V. 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). Todos os autores agradecem aos estudantes que jogaram e aos seus professores que permitiram a aplicação do experimento.

Referências

[1] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, and K. Engelhardt, "Developing Computational Thinking in Compulsory Education," 2016.

[2] White House, "President Obama Announces Computer Science For All Initiative," 2016. [Online]. Available: https://www.whitehouse.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0. [Accessed: 18-Jan-2017].

[3] A. Gomes and A. J. Mendes, "Learning to program-difficulties and solutions," in *International Conference on Engineering Education*, 2007, pp. 1–5.

[4] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, 2003.
[5] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in *19th Annual Conference on Innovation and Technology in Computer Science Education*, 2014, pp. 39–44.
[6] R. P. Borges, P. R. F. Oliveira, R. G. R. Lima, and R. W. De Lima, "A Systematic Review of Literature on Methodologies , Practices , and Tools for Programming Teaching," *IEEE Lat. Am. Trans.*, vol. 16, no. 5, pp. 1468–1475, 2018.
[7] M. Guzdial and E. Soloway, "Teaching the Nintendo generation to program," *Commun. ACM*, vol. 45, no. 4, p. 17, 2002.
[8] L. Johnson, S. A. Becker, V. Estrada, and A. Freeman, *Horizon Report: 2015 Higher Education Edition*. Austin, Texas: The New Media Consortium, 2015.
[9] D. Weintrop and U. Wilensky, "Playing by Programming: Making Gameplay a Programming Activity.," *Educ. Technol.*, vol. 56, no. 3, pp. 36–41, 2016.
[10] M. Prensky, *Digital game-based learning*. New York: McGraw-Hill, 2001.
[11] S. Arnab, S. de Freitas, F. Bellotti, T. Lim, S. Louchart, N. Suttie, R. Berta, and A. De Gloria, "Pedagogy-driven design of Serious Games: An overall view on learning and game mechanics mapping, and cognition-based models," 2012.
[12] S. Boller and K. Kapp, *Play to Learn: Everything You Need to Know About Designing Effective Learning Games*. ATD Press, 2017.
[13] T. Barnes, E. Powell, A. Chaffin, A. Godwin, and H. Richter, "Game2Learn: Building CS1 learning games for retention," in *12th Annual Conference on Innovation and Technology in Computer Science Education*, 2007, pp. 121–125.
[14] M. J. Lee, F. Bahmani, I. Kwan, J. Laferte, P. Charters, A. Horvath, F. Luor, J. Cao, C. Law, M. Beswetherick, S. Long, M. Burnett, and A. J. Ko, "Principles of a debugging-first puzzle game for computing education," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2014, pp. 57–64.
[15] K. Sung, "Computer games and traditional CS courses," *Commun. ACM*, vol. 52, no. 12, pp. 74–78, 2009.
[16] A. Vahldick, A. J. Mendes, and M. J. Marcelino, "A review of games designed to improve introductory computer programming competencies," in *44th Annual Frontiers in Education Conference*, 2014, pp. 781–787.
[17] R. Koster, *A Theory of Fun for Game Design*, 2nd ed. O' Reilly Media, Inc, 2014.
[18] M. N. A. Mazlan and L. Burd, "Does an avatar motivate?," in *41th Annual Frontiers in Education Conference*, 2011.
[19] Y. Bosse and M. A. Gerosa, "Difficulties of Programming Learning from the Point of View of Students and Instructors," *IEEE Lat. Am. Trans.*, vol. 15, no. 11, pp. 2191–2199, 2017.
[20] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.
[21] J. Sorva, V. Karavirta, and L. Malmi, "A Review of Generic Program Visualization Systems for Introductory Programming Education," *ACM Trans. Comput. Educ.*, vol. 13, no. 4, p. 15.1-15.64, 2013.
[22] M. Ben-Ari, "Visualization of programming," in *Improving computer science education*, 2013, pp. 52–65.
[23] D. Weintrop and U. Wilensky, "Bringing Blocks-based Programming into High School Computer Science Classrooms," in *Annual Meeting of the American Educational Research Association*, 2016.
[24] C. Kelleher and R. Pausch, "Lowering the barriers to programming," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 83–137, 2005.
[25] S. Xinogalos, M. Satratzemi, and C. Malliarakis, "Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment?," *Educ. Inf. Technol.*, no. 22, pp. 145–176, 2017.
[26] S. Papert, *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books, Inc., 1980.
[27] D. Laurillard, P. Charlton, B. Craft, D. Dimakopoulos, D. Ljubojevic, G. Magoulas, E. Masterman, R. Pujadas, E. A. Whitley, and K. Whittlestone, "A constructionist learning environment for teachers to model learning designs," *J. Comput. Assist. Learn.*, vol. 29, no. 1, pp. 15–30, 2013.
[28] D. Weintrop and U. Wilensky, "Situating programming abstractions in a constructionist video game," *Informatics Educ.*, vol. 13, no. 2, pp. 307–321, 2014.
[29] J. Hidalgo-Céspedes, G. Marín-Raventós, and V. Lara-Villagrán, "Playing with Metaphors: A Methodology to Design Video Games for Learning Abstract Programming Concepts," in *19th Annual Conference on Innovation and Technology in Computer Science Education*, 2014, pp. 348–348.
[30] N. R. Holbert and U. Wilensky, "FormulaT racing: Designing a game for kinematic exploration and computational thinking," in *7th International Conference on Games + Learning + Society*, 2011.
[31] D. Weintrop, N. R. Holbert, U. Wilensky, and M. Horn, "Redefining Constructionist Video Games: Marrying Constructionism and Video Game Design," in *Constructionism 2012*, 2012, pp. 645–649.
[32] C. Kazimoglu, M. Kiernan, L. Bacon, and L. Mackinnon, "Understanding computational thinking before programming: Developed guidelines for the design of games to learn introductory programming through game-play," in *Developments in Current Game-Based Learning Design and Development*, P. Felicia, Ed. Hershey, PA: IGI Global, 2013.
[33] A. L. Brown, "Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings," *J. Learn. Sci.*, vol. 2, no. 2, pp. 141–178, 1992.
[34] A. Cocciolo, "Reviewing design-based research," 2005. [Online]. Available: http://www.thinkingprojects.org/wp-content/dbr.doc. [Accessed: 14-Feb-2014].
[35] K. Gravemeijer and P. Cobb, "Design research from a learning design perspective," *Educ. Des. Res.*, pp. 17–51, 2006.
[36] DBRC, "Design-based research: An emerging paradigm for educational inquiry," *Educ. Res.*, vol. 32, no. 1, pp. 5–8, 2003.
[37] D. Walker, "Toward productive design studies," in *Educational Design Research*, J. van den Akker, K. Gravemeijer, S. McKenney, and N. Nieveen, Eds. Routledge, 2006, pp. 9–19.
[38] K. D. Squire, "Resuscitating research in educational technology: Using game-based learning research as a lens for looking at design-based research," *Educ. Technol.*, vol. 45, no. 1, pp. 8–14, 2005.
[39] B. E. Shelton and J. Scoresby, "Aligning game activity with educational goals: following a constrained design approach to instructional computer games," *Educ. Technol. Res. Dev.*, vol. 59, no. 1, pp. 113–138, Nov. 2010.
[40] G. Majgaard, M. Misfeldt, and J. Nielsen, "How design-based research and action research contribute to the development of a new design for learning.," *Des. Learn.*, vol. 4, no. 2, pp. 8–27, 2011.
[41] A. Vahldick, A. J. Mendes, and M. J. Marcelino, "Analysing the enjoyment of a serious game for programming learning with two unrelated higher education audiences," in *Proceedings of the European Conference on Games-based Learning*, 2015, vol. 2015–Janua.
[42] A. Vahldick, M. J. Marcelino, and A. J. Mendes, "Principles of a Casual Serious Game to Support Introductory Programming Learning in Higher Education," in *Gamification-based e-learning Platform for Computer Programming Education*, R. A. P. Queirós and M. T. Pinto, Eds. IGI Global, 2017.
[43] A. Vahldick, A. J. Mendes, M. J. Marcelino, and P. R. Farah, "Pensamento Computacional Praticado com um Jogo Casual Sério no Ensino Superior," in *XXIV Workshop sobre Educação em Computação*, 2016.
[44] F. L. Fu, R. C. Su, and S. C. Yu, "EGameFlow: A scale to measure learners' enjoyment of e-learning games," *Comput. Educ.*, vol. 52, no. 1, pp. 101–112, 2009.
[45] P. Sweetser and P. Wyeth, "GameFlow: a model for evaluating player enjoyment in games," *Comput. Entertain.*, vol. 3, no. 3, pp. 1–24, 2005.
[46] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York: Harper Perennial, 1990.
[47] N. Fraser, "Ten Things We ' ve Learned from Blockly," in *IEEE Blocks and Beyond Workshop*, 2015, pp. 49–50.
[48] J. Mead, S. Gray, J. Hamer, R. James, J. Sorva, C. St. Clair, and L. Thomas, "A cognitive approach to identifying measurable milestones for programming skill acquisition," in *11th Annual Conference on Innovation and Technology in Computer Science Education*, 2006, pp. 182–194.
[49] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon, 2001.
[50] A. Caspi and I. Blau, "Collaboration and psychological ownership: How does the tension between the two influence perceived learning?," *Soc. Psychol. Educ.*, vol. 14, no. 2, pp. 283–298, 2011.

Adilson Vahldick Possui doutorado em Ciências e Tecnologias da Informação pela Universidade de Coimbra (2018), mestrado em Computação Aplicada pela Universidade do Vale do Itajaí (2008) e graduação em Sistemas de Informação (2002). Atualmente é professor assistente da Universidade do Estado de Santa Catarina, em Ibirama, exercendo suas atividades docentes no curso de Engenharia de Software (Bacharelado), atuando na pesquisa e ensino de programação e desenvolvimento de jogos sérios.

Paulo Roberto Farah Possui mestrado em Informática pela Universidade Federal do Paraná (UFPR) e graduação em Bacharelado em Informática pelas Faculdades Positivo. Atualmente é professor assistente da Universidade do Estado de Santa Catarina, em Ibirama, no Departamento de Engenharia de Software, atuando no ensino, pesquisa e extensão de programação web.

Maria José Marcelino Possui doutorado em Engenharia Informática pela Universidade de Coimbra (1999). Atualmente é Professora Auxiliar na mesma universidade, no Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia, onde ensina programação e simulação e investiga nas áreas do ensino da programação, da simulação e do Ensino a Distância.

António José Nunes Mendes Possui doutorado em Engenharia Informática pela Universidade de Coimbra. É Professor Associado do Departamento de Engenharia Informática da mesma universidade, lecionando essencialmente unidades curriculares de programação. Investiga nas áreas do ensino da programação e do ensino a distância. Coordena o Projeto de Ensino a Distância da UC.