ProgBotics: jogo para estímulo da prática de programação

André Vasconcelos[1]* Igor Knop[2]

[1]Universidade Federal de Juiz de Fora, Departmento de Ciência da Computação, Brazil

Figura 1: Recorte da tela da primeira fase do ProgBotics, jogo para estímulo da prática de programação.

RESUMO

O estudo de programação de computadores é considerado complexo e conhecido por exigir um alto grau de abstração e pensamento crítico na resolução de problemas. Adicionalmente, exige um grande tempo de prática com as linguagens de programação para que as mesmas não imponham barreiras na busca para a resolução de problemas. Isso conduz a um alto índice de abandono e reprovação nas disciplinas básicas de computação em instituições que as disponibilizam nos anos iniciais dos cursos de exatas e engenharias. Entre as abordagens para estimular os alunos a praticarem mais a programação, está o uso de jogos e competições. Este trabalho apresenta o *ProgBotics*, um jogo multiplataforma no qual o objetivo é apresentar aos estudantes problemas de programação na forma de quebra-cabeças com simulação de física em tempo real. Os desafios são resolvidos através de uma linguagem de programação simplificada que move três robôs em cenários previamente estabelecidos. Durante a execução, o desempenho do aluno é registrado e medido e estudos subsequentes podem ser realizados para categorizar habilidades ou detectar carência de atendimento diferenciado.

Keywords: Jogos digitais, linguagens de programação, avaliação de ensino.

1 INTRODUÇÃO

O melhor entendimento das especificidades do ensino de programação e a posterior melhoria dos processo podem contribuir para um aumento das aprovações nas disciplinas de programação e por consequência, aumentar a eficiência na aplicação dos recursos públicos e privados no ensino e a diminuir a evasão de alunos.

Este trabalho, ainda em desenvolvimento, busca observar o engajamento do aluno através da prática de programação. Para tanto, é proposto um jogo digital multiplataforma chamado *ProgBotics* que usa os conceitos de básicos de programação para apresentar uma boa experiência de entretenimento ao aluno. Espera-se que através da prática lúdica constante o aluno tenha um complemento ao material apresentado dentro das disciplinas iniciais.

*e-mail: andre.vasconcelos@ice.ufjf.br

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Esta seção faz uma breve revisão dos trabalhos que buscam a melhoria do processo de ensino de programação por uma série de abordagens diferentes.

2.1 O ensino de programação

O estudo de algoritmos muitas vezes é considerado complexo ou desmotivante para quem inicia os estudos nas disciplinas básicas de computação. Estima-se que a média de aprovação nas disciplinas básicas de programação fica entre 60 e 70% independente da linguagem de programação e país de origem [11, 17, 12].

Entre os problemas observados em algumas instituições, destacam-se a falta de aconselhamento na escolha do curso, falta de proficiência em matemática e resolução de problemas, problemas de planejamento nas disciplinas iniciais de programação, falta de prática e retorno, mau gerenciamento dos estudos e natureza das linguagens de programação utilizadas [11, 15, 16].

Entre as medidas para minimizar o problema, estão a criação de oficinas de estudo[14], novas abordagens didáticas das disciplinas ou do uso de jogos como forma de estímulo ao estudo. Este último método é de especial interesse pela forte aceitação entre os alunos durante o processo de aprendizagem [13] e por associá-lo a uma atividade de lazer em todas as idades [13, 10].

2.2 Jogos que utilizam programação

Outros jogos já utilizam a programação como mecanismo básico para construir a experiência de jogo. Um exemplo disso é o *Doodle* de comemoração de 50 anos de programação para crianças [3], um jogo disponibilizado pela Google que, no dia que foi lançado, comemorava o ensino de algoritmos para crianças.

O *LightBot*[6], que motivou este trabalho, permite programar um robô cujo objetivo é iluminar espaços em um tabuleiro. Ele já é utilizado como complemento nas aulas iniciais de nossa instituição de ensino e tem um retorno positivo por parte dos alunos. Das suas

principais características pode-se destacar a linguagem simples de programação e intuitiva redução do escopo de problemas a um tema específico. A execução do programa é realizado em turnos discretos que refletem na posição do robô no tempo.

O *Human Resources Machine*[4] é um jogo comercial com estruturas mais complexas de programação como filas, saída de dados, laços e etiquetas de desvio. Ele desafia o jogador a passar por codificações realmente complexas para resolver os desafios. O principal atrativo estão nos gráficos e no público alvo de jogadores que querem desafios realmente complexos (muitas vezes as respostas finais se tornam quase ilegíveis à primeira vista).

O *RoboCode*[8] trata-se de um jogo competitivo no qual seus usuários criam códigos que geram inteligências artificiais para pequenos tanques de guerra. Com as inteligências artificiais geradas, os tanques se enfrentam em batalhas diversas e ganha o que estiver seguindo as instruções dos melhores algoritmo.

Fora dos jogos digitais, o jogo de tabuleiro *RoboRally*[7] foi um dos primeiros jogos de mesa no qual robôs são programados de forma competitiva para completarem um percurso em um ambiente hostil e competitivo.

Pela própria sequencialidade de execução de um código, esses jogos no geral empregam um sistema de grade e "turnos" para a execução dos comandos. Tornando o movimento dos robôs discreto. O ProgBoticsse difere-se por permitir um movimento contínuo, mesmo que a programação seja discreta. Isso resulta em posições fora da grade do cenário quando os robôs colidem com algum obstáculo ou entre eles.

3 MÉTODO

O projeto se encontra na primeira de três fases distintas: a implementação do jogo digital ProgBotics; implementar um sistema de telemetria para acompanhamento do uso e progresso dos alunos dentro de um período de tempo; e a condução de um experimento para identificar a relacionação com o perfil dentro do jogo com o desempenho na disciplina e com um grupo de controle.

3.1 Implementação do protótipo

O `ProgBotics` foi implementado através da linguagem de programação *Java*, com o *framework* libGDX[5] e com ambiente de desenvolvimento *Android Studio*[1]. Adicionalmente, a ferramenta *Tiled* [9] foi utilizada na produção dos cenários e as imagens foram obtidas de um banco de imagens com licença de uso *Creative Commons* para representar os elementos do jogo. Toda a simulação de física de corpos é gerida pela biblioteca *Box2d*[2] através da *libGDX*. O jogo pode ser exportado como um aplicativo Android, uma aplicação Java para computadores pessoais e como uma aplicação HTML5 para navegadores.

Os controladores devem se manter mais próximos aos elementos do jogo do ponto de vista temático como sensores de luz, transmissores e cartões de memória simulados dentro do jogo. Acredita-se que ao manter essa abordagem, não serão perdidos os elementos lúdicos esperados.

3.2 Uso do ProgBotics

Ao iniciar o jogo, é apresentada uma sala na qual pode-se ver robôs coloridos e monitores correspondentes a cada um destes. Na tela também são disponibilizados comandos que permitem a programação dos movimentos para cada um dos robôs. O objetivo de todos os cenários consiste em fazer cada um dos autômatos tocar no seu respectivo terminal.

Para atingir o objetivo, é necessário programar os robôs com os comandos disponibilizados. Cada comando faz com que os robôs caminhem em uma determinada direção por um determinado tempo (meio segundo). O jogador tem uma quantidade limitada de tempo para fazer os robôs chegarem nos terminais ou ele falha no cenário e tem que reiniciar o cenário. Caso consiga atingir o objetivo dentro do tempo limite, o cenário seguinte se inicia com um novo desafio em dificuldade crescente.

3.3 Etapas de Programação e Execução

Cada cenário do *ProgBotics* possui duas etapas: a de Programação e a de Execução. Os cenários sempre iniciam com a etapa de Programação. Nela, os comandos de movimentação básicos (andar para uma das quatro direções do eixo cartesiano ou ficar parado) são adicionados pelo jogador para cada um dos robôs. Dois botões são disponibilizados e permitem a escolha entre os robôs disponíveis de forma que o jogador possa visualizar e alterar a programação de cada um deles.

A segunda etapa de cada cenário é a de Execução. Nesta etapa, os comandos são executados em cada um dos robôs. Durante a execução eles não podem mais ser editados. Com a ativação, os robôs se locomovem pelo cenário de acordo com a programação criada para eles. Quando um robô colide com algum elemento, sua posição (ou a do outro elemento) é alterada. Se o elemento é um monitor correspondente ao robô, o dado elemento é ativado.

É durante a execução que a experiência é construída: pela dificuldade de se programar três ou mais robôs ao mesmo tempo, o aluno é desafiado a realizar um planejamento antes da execução e a realizar uma análise do resultado. Como os robôs interagem mutuamente, um defeito no planejamento causa a interferência no resultado do grupo.

Atualmente o conjunto de cenários possui o mesmo limite de tempo (15 segundos) para sua conclusão e três pares robôs-terminais. Entretanto, é possível ter esses parâmetros alterados em cenários futuros. Os cenários do *ProgBotics* se tornam mais desafiantes à medida que a interação entre os robôs e os obstáculos se torna mais complexa.

3.4 Montagem dos cenários

Cada cenário é desenvolvido utilizando a ferramenta *Tiled* com um conjunto de ladrilhos padrão. Eles são formados por quatro camadas de objetos que controlam a disposição dos elementos do jogo e mais algumas camadas são utilizadas apenas para desenho. A Figura 2 apresenta o primeiro cenário na ferramenta.

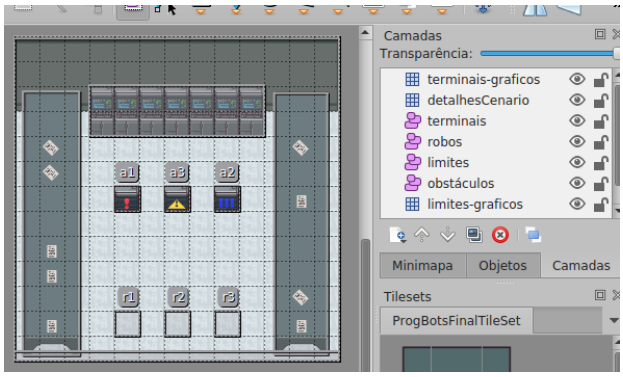

Figura 2: Recorte do primeiro cenário feito na ferramenta Tiled: os elementos do jogo são criados por camadas de objetos.

A primeira camada é utilizada para o desenho do cenário e possui elementos para compor a sua atmosfera. Uma segunda camada de objetos define os limites físicos para a simulação, não permitindo que os robôs e demais elementos a atravessem.

A terceira camada de objetos define a posição de cada um dos robôs e as cores são associadas pela ordem que aparecem nela (atualmente há quatro esquemas de cor para os robôs do jogador e um para personagens não jogáveis).

A quarta camada possui a localização e dimensão de cada terminal. Também são utilizados três esquemas de cores e cada um é associado a um robô na camada anterior de mesmo índice. Por fim, a quinta camada possui os objetos de cenário que os robôs podem empurrar, alterando suas trajetórias.

3.5 Modelo de dados

Na leitura do cenário, cada elemento lido da camada de objetos de robôs é instanciado como um objeto `Personagem` que possui como propriedades um corpo físico (instância de `com.badlogic.gdx.physics.box2d.Body`) que guarda sua posição, massa e figura geométrica utilizada (atualmente um quadrado). Cada `Personagem` também possui uma lista de comandos texto associados à constantes para os movimentos. Um índice serve como contador de programa para saber qual a próxima instrução será executada.

Cada instância de `Terminal` também possui um corpo físico com uma forma quadrada mas é um elementos estático: não pode ser deslocado. Cada terminal também possui uma referência para um único personagem que o ativa. Essa associação é feita durante a instanciação pelo robô de mesmo índice durante o carregamento do cenário. Corpos físicos estáticos também são gerados para os limites do cenário, para impedir que os robôs se movam para fora da área esperada.

A interação do usuário com o jogo é capturada dentro do jogo através de instâncias de `Registro` gerenciadas pela instância geral de controle do jogo. Atualmente, são registrados três pontos com data e hora: quando se iniciou um cenário; quando se iniciou a execução; e quando houve uma conclusão do cenário.

Adicionalmente, ainda nas instâncias de `Registro` são armazenados o nome do cenário, um identificador gerado para o usuário (ainda não há um sistema de autenticação) e um texto com o resultado final do cenário ("sucesso", "falha" ou "interrompido"). Os dados são capturados localmente, mas o servidor usado para a coleta e análise desses dados ainda está em desenvolvimento.

4 Resultados

A implementação do ProgBoticspela libGDX permitiu a geração de um jogo multiplataforma com uma experiência equivalente em três plataformas diferentes. Para os dispositivos móveis foi possível realizar a geração de um aplicativo para o sistema Android, mas não para o iOS por dificuldades de configuração do ambiente de desenvolvimento. A versão para computadores pessoais utiliza a mesma estrutura do aplicativo, mas é executada dentro de um ambiente Java e não apresentou problemas. A versão para HTML se mostrou mais problemática, devido às dificuldades na compilação do jogo através de GWT, o que impossibilitou uma boa e aceitável experiência nos navegadores.

A Tabela 1 apresenta os dados colhidos de um usuário que passa por quatro cenários. São medidos, em segundos, nas três últimas colunas: o tempo entre a apresentação do desafio à execução; o tempo da execução à falha ou sucesso do cenário; e o tempo total da apresentação ao resultado.

Tabela 1: Trecho de dados colhidos durante a execução. Todos os dados estão em segundos.

Cenário	Status	Programando	Executando	Tentativa
Level2	sucesso	11	4	15
Level2b	falha	48	14	62
Level2b	sucesso	27	8	35
Level3b	falha	135	15	150
Level3b	sucesso	18	7	25
Level4	falha	65	15	80
Level4	sucesso	56	14	70

Os dados de utilização puderam ser colhidos mas só puderam ser observados pela depuração local já que o envio para o servidor ainda está para ser realizado brevemente. Através da centralização da coleta dos dados, espera-se poder fazer uma análise mais apurada e levantamento de perfil dos utilizadores.

5 Considerações Finais e Perspectivas Futuras

Aproximadamente um terço dos alunos ingressantes nos cursos básicos de programação são reprovados ou abandonam a disciplina. Este trabalho apresenta a primeira etapa de projeto para a melhoria no processo de ensino de programação na forma de um jogo digital. O ProgBoticsé um jogo digital multiplataforma baseado em programação que se propõe a servir de apoio educacional e ferramenta de acompanhamento de desempenho.

O ProgBoticsencontra-se em pleno desenvolvimento, com a versão atual com a programação de robôs completamente funcional. Estes robôs se movem livremente em um espaço que possui simulações físicas em seu ambiente. Um conjunto de mapas iniciais foi desenvolvido e são oferecidos em um ordem crescente de complexidade. Os comandos são mantidos simples por projeto para aproximar o tema e reduzir o nível de abstração na resolução dos problemas exigido do aluno.

Através da programação dos robôs, os alunos são capazes de planejar e confrontar seus resultados, observando se seus objetivos foram atingidos ou se seus algoritmos precisam de alterações. Os dados são colhidos localmente para posteriormente traçar o perfil de uso do usuário. Entretanto, a análise desses dados ainda é limitada pois a sua submissão para um servidor central ainda está em desenvolvimento. De posse desses dados de forma centralizada, espera-se futuramente acompanhar o envolvimento dos alunos e cruzar estes dados com os resultados da disciplina

Por fim, os autores agradecem à Universidade Federal de Juiz de Fora, CNPq e Capes pelo apoio neste trabalho, fruto de um projeto de treinamento profissional na graduação.

Referências

[1] Android Studio. `https://developer.android.com/studio`. Accessed: 2018-09-19.

[2] Box2d. `http://box2d.org`. Accessed: 2018-09-19.

[3] Doodle de comemoração de 50 anos de programação para crianças. `https://www.google.com/doodles/celebrating-50-years-of-kids-coding?hl=pt-BR`. Accessed: 2018-09-19.

[4] Human Resource Machine. `https://tomorrowcorporation.com/humanresourcemachine`. Accessed: 2018-09-19.

[5] libGDX. `https://libgdx.badlogicgames.com`. Accessed: 2018-09-19.

[6] Lightbot. `http://lightbot.com/`. Accessed: 2018-09-19.

[7] Robo Rally. `http://avalonhill.wizards.com/games/robo-rally`. Accessed: 2018-09-19.

[8] Robocode. `https://robocode.sourceforge.io/`. Accessed: 2018-09-19.

[9] Tiled Map Editor. `http://www.mapeditor.org`. Accessed: 2018-09-19.

[10] D. A. Batista and C. L. Dias. O processo de ensino e de aprendizagem através dos jogos educativos no ensino fundamental. In *Revista Colloquium Humanarum, São Paulo*, volume 9, pages 975–982, 2012.

[11] T. Beaubouef and J. Mason. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106, 2005.

[12] Y. Bosse and M. A. Gerosa. As disciplinas de introdução à programação na usp: um estudo preliminar. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1389, 2015.

[13] C. J. Haguenauer, F. Carvalho, A. L. Q. Victorino, M. Lopes, and F. Cordeiro Filho. Uso de jogos na educação online: a experiência do latec/ufrj. *Revista Educa online, UFRJ*, 1(1):1–14, 2007.

[14] J. Júnior, C. E. Rapkiewicz, C. Delgado, and J. A. M. Xexeo. Ensino de algoritmos e programação: uma experiência no nível médio. In *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil*, 2005.

[15] W. Priesnitz Filho, I. Abegg, and E. de Oliveira Simonetto. Uma abordagem diferenciada no ensino de algoritmos através da utilização de uma lousa digital. *Revista GEINTEC-Gestão, Inovação e Tecnologias*, 2(2):129–137, 2012.

[16] A. Santos, A. Gorgônio, A. Lucena, and F. Gorgônio. A importância do fator motivacional no processo ensino-aprendizagem de algoritmos e lógica de programação para alunos repetentes. In *WEI-Workshop sobre Educação em Computação*, pages 1–10, 2015.

[17] C. Watson and F. W. Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 39–44, New York, NY, USA, 2014. ACM.