Maze Code: Retórica Procedural Aplicada ao Ensino de Lógica de Programação

Jose Grigorio Neto
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus UFV-Florestal
Florestal, Brasil
jose.grigorio@ufv.br

Paulo Henrique Pimentel Marcolino
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus UFV-Florestal
Florestal, Brasil
paulo.marcolino@ufv.br

Pablo Ferreira
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus UFV-Florestal
Florestal, Brasil
pablo.ferreira@ufv.br

Daniel Mendes Barbosa
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus UFV-Florestal
Florestal, Brasil
danielmendes@ufv.br

***Resumo*—Considerando que reprovações em disciplinas introdutórias à programação são um problema comum em graduações na área da computação, e, tendo em vista a popularização dos jogos digitais e os benefícios que eles podem trazer à sociedade, este trabalho consiste na apresentação de um protótipo de um jogo educativo, chamado Maze Code, que tem como foco o ensino de conceitos envolvendo programação, tendo como base na sua construção a retórica procedural e o design participativo. O protótipo foi avaliado por alunos do curso de ciência da computação da Universidade Federal de Viçosa - Campus Florestal e os resultados obtidos reforçam que a retórica procedural é eficaz para o ensino de programação básica no contexto do trabalho. Também foi possível observar que o design participativo ajudou a melhorar a aceitação do jogo pelos alunos do curso.**

***Palavras-chave*—jogo educativo, programação, retórica procedural, design participativo, jogos sérios, compiladores, interpretadores**

I. Introdução

Há uma grande taxa de reprovações em disciplinas introdutórias à programação em cursos superiores na área de computação. Um levantamento dos resultados das disciplinas de introdução à programação da Universidade de São Paulo entre 2010 e 2014 mostra que em média 30% dos alunos são reprovados nestas disciplinas [1]. A reprovação em uma disciplina introdutória pode atrasar a formação do aluno e pode até mesmo levar à evasão precoce do curso.

Jogos sempre estiveram presentes na história da humanidade, tanto para diversão quanto para a educação e aprendizado, estimulando o raciocínio, a estratégia, a criatividade e outros aspectos físicos e mentais. A utilização de jogos com um viés educacional vem crescendo [2], de forma a envolver o jogador em um ambiente mais lúdico e interativo, transformando o assunto em uma atividade mais divertida, prazerosa e envolvente.

Os computadores tem ocupado um espaço relevante na vida das pessoas, sendo inclusive um meio de comunicação, trabalho e lazer para muitas delas. Crianças e adolescentes são apresentados ao uso de dispositivos tecnológicos como computadores e *smartphones* desde cedo, e de acordo com as estimativas mundiais, 97% dos jovens se dedicam a jogos de computador ou consoles [3].

Os jogos digitais vêm ganhando cada vez mais espaço na vida de todos [4], com o aumento do interesse dos jogadores e o tempo em que se dedicam aos jogos. Parte disso se deve ao avanço tecnológico que possibilitou uma maior qualidade, complexidade e estruturação dos jogos, além da sua disponibilização em vários formatos, sejam eles plataformas móveis, consoles ou até nos computadores tradicionais.

Este trabalho teve como objetivo desenvolver um jogo sério com temática de programação para auxiliar os novos alunos do curso de ciência da computação da Universidade Federal de Viçosa - Campus Florestal no decorrer da disciplina de programação, onde alguns deles terão seu primeiro contato com programação. O jogo considera o uso da retórica procedural ao apresentar problemas computacionais que devem ser resolvidos utilizando uma mecânica de programação em blocos.

O restante do trabalho foi organizado da seguinte forma: a Seção II apresenta os conceitos utilizados na construção do jogo; a Seção III apresenta alguns jogos com temática relacionada; a Seção IV os materiais e métodos utilizados; a Seção V apresenta o Maze Code; a Seção VI apresenta os resultados da avaliação do protótipo criado e a Seção VII as conclusões.

II. fundamentação teórica

A utilização de jogos como um método didático é um tema importante, uma vez que essa utilização tem o objetivo de aumentar qualitativamente a aprendizagem e o ensino de uma área ou conteúdo [5], [6], [7]. Apesar de ser um tema que vem sendo pesquisado recentemente, ainda possui algumas lacunas e oportunidades para se criar e desenvolver novas ferramentas para imersão, junto a novas estratégias de como o

jogo é propriamente construído e estruturado, e para contextos específicos ou o ensino de um assunto específico.

Neste contexto, o presente trabalho apresenta o jogo Maze Code, um *serious game* do gênero *escape room* que tem como objetivo auxiliar na aprendizagem de programação bem como dos conceitos relacionados e também contribuir para o aumento do interesse na área, sendo construído com base no conceito da retórica procedural. Por se tratar de um jogo com a temática de programação, conceitos de compiladores e interpretadores de linguagens de programação foram utilizados durante seu desenvolvimento.

Inicialmente, durante o ano de 2020, foi disponibilizada uma primeira versão estável do jogo para alunos do curso com o objetivo de encontrar erros e recolher sugestões. E em 2021, para que fosse obtido um resultado sólido, foi disponibilizada a segunda versão estável do Maze Code novamente para os alunos do curso para obter um feedback sobre as melhorias feitas no jogo, encontrar novos erros e recolher novas sugestões.

A. Serious Games

Jogos sérios ou *serious games* são jogos utilizados com o propósito de ensino, aprendizagem ou treinamento, ou seja, são ferramentas para simular problemas do mundo real num mundo digital [8]. Os Jogos sérios são utilizados numa variedade de áreas, como, por exemplo: educação, treinamento profissional, saúde, publicidade e políticas públicas.

B. Compiladores

Compilador [9] [10] é um tipo de programa que tem como entrada um código escrito em alguma linguagem de programação fonte, e produz um código em uma outra linguagem objeto, como a linguagem de máquina, por exemplo.

C. Interpretadores

Interpretadores [9] são derivados dos compiladores, e são sistemas que tem como entrada um código de uma linguagem de programação fonte e conseguem executá-lo, sem a necessidade de traduzi-lo para linguagem de máquina. Portanto, o código-fonte é interpretado à medida que ocorre a execução, eliminando a etapa de produzir um código para a linguagem objeto, como os compiladores fazem [11].

D. Retórica Procedural

A retórica procedural consiste em identificar e estudar os argumentos criados por processos, ou seja, analisar que tipo de mensagens são passadas por determinados processos de um jogo e utilizá-las para criar jogos que expressam argumentos por meio de seus processos. Desta forma, ao longo do jogo o jogador passa a entender o contexto e as suas necessidades sem precisar de ajuda, dando uma sensação maior de aprendizado [12].

E. Design Participativo

O Design Participativo é uma abordagem que envolve todas as partes interessadas no projeto para ajudar em seu design [13], assim garantindo que seu resultado atenda às expectativas de todos os envolvidos, em especial daqueles que são historicamente excluídos dos processos de decisão, como, por exemplo, o público-alvo.

F. Imersão e narrativa

Segundo Mendonça e Mustaro [14], um dos pontos fundamentais que se deve levar em consideração durante a criação de um *serious games* é a utilização de elementos de imersão e narrativa para manter a motivação do estudante. Dessa forma, os autores propõem a utilização de algumas estratégias visando alcançar essa motivação por parte dos estudantes. São elas: i) apresentar uma estrutura do universo do *serious games* que represente um certo tipo de semelhança com o ambiente onde o estudante está inserido. Isso pode ser realizado explorando o tema, a trama e também a escolha do personagem. ii) utilização de elementos que consigam despertar um vínculo emocional por parte do estudante ao *serious game*. iii) apresentar uma narrativa que consiga envolver o estudante dentro dela. iv) escolher uma forma de contar histórias de forma que haja uma evolução da narrativa e do personagem junto com a evolução do estudante a medida que ele explora os desafios do *serious game*. Outro fator importante para garantir a proximidade do jogo com o mundo real é a utilização de elementos gráficos e sonoros que tragam um conforto e uma sensação agradável ao estudante [15]. Seguindo essa ideias, construímos o Maze Code com alguns elementos de imersão e narrativas visando trazer uma experiencia lúdica junto com a proposta de um *serious game*.

III. Trabalhos relacionados

A seguir são citados alguns trabalhos com propostas parecidas com a proposta do Maze Code, como jogos sérios com propósito educacional, voltados à introdução de um assunto até então novo ao jogador, que estejam ligados à área da computação ou matemática e que tenham estudantes como principal público-alvo.

O Operação Lovelace [16] é um jogo de estratégia 2D, voltado para o público infantil (de 8 a 11 anos), onde são apresentados conteúdos indicados pelo Currículo de Tecnologia e Computação para estudantes do 5º ano do Ensino Fundamental: listas, filas e pilhas. A narrativa do jogo é apresentada em formato de *storyboard* de forma lúdica ao jogador, apresentando as personagens principais, duas cientistas que precisam do auxílio do jogador para ajudar o mundo com suas criações. O jogo se baseia em uma mecânica de programação em blocos para realizar a movimentação do seu personagem, para assim, chegar ao objetivo de cada fase enquanto evita alguns inimigos e obstáculos.

O jogo As Aventuras Espaciais de Cody [17] tem como objetivo ensinar e treinar noções iniciais de lógica de programação. A história se passa em um mundo fictício onde o planeta natal do jogador vive uma crise financeira. O jovem

Cody então é eleito para partir em uma jornada pelo espaço em busca de moedas intergaláticas para salvar seu planeta da crise financeira. Cada fase do jogo é um planeta novo para Cody explorar e durante cada uma delas o jogador deverá escrever um código de programação descrevendo como o personagem deve se movimentar pelo ambiente recolhendo todas as moedas intergaláticas disponíveis.

Em Mundo de Euclides [18], um jogo 2D plataforma, o enredo se baseia em Juninho, o personagem principal, que estava estudando para uma prova de Geometria Espacial quando acaba caindo no sono. Enquanto dorme, Juninho surge no mundo de Euclides, e presencia o vilão do jogo capturando Euclides, o pai da geometria. O objetivo de Juninho passa a ser salvar Euclides para sair daquele mundo. No início de cada fase o jogador é apresentado a um axioma matemático diferente e deve seguir instruções baseadas nesse axioma para cumprir a missão referente à fase.

O Maze Code se diferencia dos trabalhos citados acima em alguns aspectos: i) a mecânica principal dos desafios do jogo, onde o jogador forma códigos simples de programação ao combinar os blocos coletados; ii) a não-linearidade da progressão do jogo, onde o jogador escolhe por qual caminho seguir; iii) a possibilidade de exploração do mapa, visto que os jogos citados possuem fases bem definidas, não deixando espaço para a exploração; iv) mesmo o público-alvo sendo composto por estudantes, a faixa etária indicada é diferente, já que o Maze Code é voltado para universitários.

IV. Metodologia

A. O Unity

O Unity[1] é uma das *engines* gráficas mais famosas do mercado atual, principalmente por conta de sua variedade de ambientes de criação, por sua simplicidade de desenvolvimento e pela ampla lista de plataformas suportadas como: ios, Android e Windows Phone para mobile; Windows, Linux e Mac para computadores e Xbox One, Nintendo Switch e Playstation 4 para consoles.

O Unity possui um modo de desenvolvimento voltado apenas para o desenvolvimento de jogos 2D, no qual é possível manipular os objetos de maneira mais simplificada, além de contar com um editor de *sprites* e ferramentas específicas para jogos 2D, como um motor de física especial para lidar com a física 2D, utilizando otimizações disponíveis apenas em 2D. Todas estas características justificam nossa escolha em utilizar o Unity no desenvolvimento deste jogo.

B. Testes e Formulários

Ao longo do desenvolvimento deste projeto, foram disponibilizados questionários de acordo com o método de questionários anônimos, um dos métodos citados em [19], aos alunos do curso, sendo o primeiro questionário aplicado aos alunos ingressantes do curso em 2019 com o objetivo de avaliar a relevância e a aceitação do presente jogo que estava em planejamento.

[1]Unity - https://unity3d.com/pt/unity

A partir das Figuras de salas e telas de interface construídas para a apresentação do conceito do jogo, foi elaborado um questionário com o objetivo de avaliar a relevância e a aceitação do presente jogo pelo seu público-alvo. Este questionário foi aplicado a uma turma inicial do curso Bacharelado em Ciência da Computação. O questionário é composto por questões referentes ao protótipo do jogo propriamente dito e dos conceitos utilizados.

Após termos a primeira versão beta do Maze Code, foi aplicado um segundo questionário a estudantes de todos os períodos a fim de conseguir dados sobre *bugs* e melhorias que poderiam ser adicionadas ao jogo. O mesmo foi composto apenas por uma Seção formada também pela identificação do voluntário e com uma questão aberta referente à experiência com o jogo.

Houve outras duas aplicações de questionários. Um mesmo modelo de questionário foi aplicado duas vezes em momentos diferentes do desenvolvimento do jogo: um após o desenvolvimento da primeira versão estável, em 2020, e outro em 2021 após a segunda versão ser desenvolvida. Esse questionário era composto por quatro seções: a primeira delas formada por perguntas básicas sobre faixa etária e tempo de jogo gasto no Maze Code; a segunda Seção continha perguntas sobre a satisfação com o jogo em geral com uma escala de 1 a 5 para medição de opiniões; e a terceira Seção por questões discursivas sobre sugestões referentes ao jogo, *design* e história do jogo.

Junto desses questionários, os alunos recebiam um outro questionário opcional que tinha o objetivo de coletar relatos de *bugs*, onde o aluno poderia enviar fotos e vídeos mostrando qual erro aconteceu e onde aconteceu.

V. O Jogo

A. Estética

Fig. 1. Tela inicial do jogo.

Tendo como essência o ensino de programação, o desenvolvimento do Maze Code seguiu algumas etapas cruciais para a sua construção. A primeira etapa foi a definição da plataforma de desenvolvimento e suas dependências que já foram explicados anteriormente. A etapa seguinte foi a definição do gênero (*Escape Room*) e por fim a estética do jogo.

Como o jogo tem como base o estilo *escape room*, o jogador assume o controle do personagem principal em um mundo de duas dimensões (2D), com total liberdade para se movimentar, onde a única ação possível de ser realizada é interagir com o ambiente e com os itens alocados no mesmo. Logo, a jogabilidade se dá na exploração de salas e labirintos e na interação com os itens presentes nesses ambientes. O jogador avança e progride no jogo à medida que, interagindo com esses itens, recebe blocos de programação, que serão utilizados para resolver os quebra-cabeças que lhe são apresentados, criando um algoritmo que satisfaça os requisitos e que seja capaz de gerar uma possível solução, apenas com os blocos que o jogador possui em seu inventário. Após definir toda a questão estética do jogo, foi criada então a tela inicial do jogo, mostrada na Fig. 1.

B. Trilha sonora

Pensando nos conceitos apresentados por [15], foi utilizado para a criação da trilha sonora a ferramenta Beepbox[2] e com ela criamos uma trilha sonora principal que permanece ativa durante toda a jornada no jogo e também várias trilhas sonoras que são disponibilizadas durante a resolução dos desafios e são habilitadas à medida que o jogador avança no jogo.

C. História

Para a história, criamos um roteiro onde está descrito o enredo, história, personagens e diálogos presentes no jogo. A narrativa foi estruturada de forma que o estudante se sinta envolvido com o jogo. A história gira em torno de um estudante chamado Lucas, que finalizou o ensino médio e está em busca de uma bolsa de estudos em cursos de computação, e para conseguir essa bolsa, ele entra em uma competição de lógica de programação onde ele deve superar seus medos e desafios e completar o labirinto Maze Code. Dentro desse labirinto ele irá encontrar dois personagens amigos importantes, Yago e Isabella, que ele irá ajudar e ser ajudado ao longo do labirinto. Além disso, o protagonista irá encontrar alguns professores espalhados pelo labirinto que irão premiá-lo com emblemas, onde cada um desses emblemas é entregue após o professor explicar sobre pontos importantes de um bom programador. Durante os diálogos com os amigos, o protagonista irá expor sentimentos de medo e insegurança que serão combatidos e vencidos ao longo da narrativa, de forma que o estudante que estará jogando possa se identificar com o protagonista e possa criar um vínculo emocional que o fará permanecer jogando para concluir a narrativa e consequentemente os desafios.

D. Problemas

Os problemas encontrados durante o jogo devem ser resolvidos em um ambiente gráfico chamado terminal. Nele, o jogador poderá ligar blocos de programação formando um código que quando executado deve resolver aquele problema específico. Para entrar em um terminal basta interagir com os computadores que estão espalhados pelo mapa, onde cada computador está ligado a uma porta e caso o jogador consiga resolver aquele problema a porta se abrirá.

Os problemas de modo geral são problemas comuns de serem resolvidos em disciplinas introdutórias de programação, como o exemplo do primeiro problema do jogo, mostrado na Fig. 2. O objetivo desta estratégia é proporcionar ao jogador uma forma de reconhecer alguns aspectos da disciplina dentro do jogo, a fim de ajudá-lo a reconhecer os problemas resolvidos no jogo e resolvê-los também na vida real.

Fig. 2. Exemplo da apresentação de um simples quebra-cabeça.

A resolução desses se dá por meio de uma construção lógica utilizando os blocos disponíveis no inventário do jogador. Um exemplo de solução viável para um problema pode ser visto na Fig. 3, que apresenta a solução para o problema da Fig. 2.

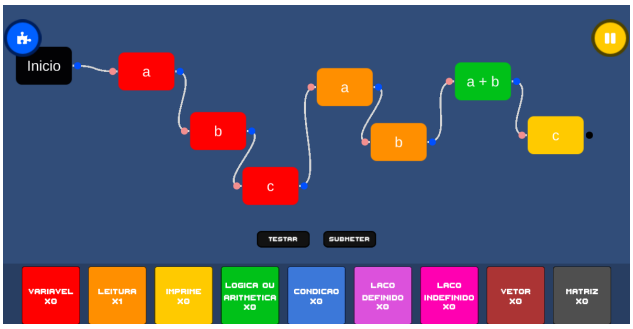

Fig. 3. Uma solução possível para o problema da primeira sala.

Caso o jogador não possua blocos o suficiente para resolver um problema, ele será impedido de interagir com o terminal e uma mensagem aparecerá explicando o motivo como pode ser visto na Fig. 4. Existem três casos onde é possível que o jogador se encontre nessa situação: o primeiro caso é onde o jogador ainda não explorou todo o mapa possível até aquele momento, ou seja, ele deverá entrar em salas ainda não visitadas em busca de mais blocos; o segundo caso é quando o nível da sala é incompatível com os blocos liberados pelo jogador, sendo que neste caso outros terminais estarão disponíveis e, depois de liberar um novo bloco, o jogador poderá voltar até este terminal específico; o último caso é quando o jogador acabou gastando mais blocos que o necessário para resolver alguns problemas e acabou ficando sem blocos suficientes para avançar no jogo, mas neste caso

[2]Beepbox - https://www.beepbox.co/

são gerados novos blocos em locais aleatórios no mapa, e o jogador então deverá voltar às salas já visitadas em busca de novos blocos.

Fig. 4. Quando o jogador não possui os blocos necessários.

E. O mapa

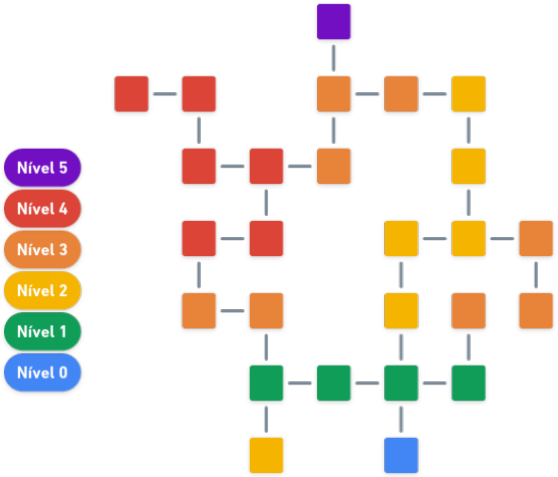

Fig. 5. Mapa do jogo

O mapa do Maze Code, mostrado na Fig. 5, é um labirinto com 26 salas e para entrar em cada uma delas o jogador deverá resolver problemas variados. Como visto na Tabela I, os problemas são divididos em 5 níveis de dificuldade e pensados para reforçar o aprendizado do estudante gradativamente com o uso da Retórica procedural.

TABELA I
TABELA DE NÍVEIS DE DIFICULDADE

Níveis	Abordagem
Nível 1	Aritmética básica
Nível 2	Comandos if e else
Nível 3	Estruturas de repetição
Nível 4	Vetores e matrizes
Nível 5	Dificuldade aumentada

Os problemas de nível 1 servem para introduzir o jogador ao ambiente do jogo e funcionam como um tutorial progressivo, onde a cada um dos problemas solucionados o jogador aprende algo novo sobre as mecânicas básicas do jogo. Os problemas de nível 2 encerram esse período de adaptação e começam a cobrar maior complexidade nas resoluções.

A função dos problemas de nível 3 é de consolidar o conhecimento a respeito do que já foi ensinado até aqui e trazer uma introdução a novos assuntos, como os comandos de repetição, fazendo com que seja difícil um jogador iniciante avançar para uma fase de nível 3 antes de terminar todas as dos outros níveis.

No nível 4, o jogo espera que o jogador já tenha entendido todos os conceitos de programação e esteja preparado para os desafios complexos.

O nível 5 é o nível final, onde tudo aprendido será testado em apenas uma sala. O jogador deverá resolver um desafio maior que qualquer outro resolvido até então. Caso consiga resolvê-lo, a porta de saída do labirinto se abre e o jogador poderá sair, terminando o jogo.

O jogador poderá explorar todo o mapa do jogo, porém as entradas das salas costumam estar bloqueadas. Para desbloquear uma sala, o jogador precisa resolver um problema no terminal mais próximo à porta da sala que está bloqueada. É esperado que o jogador aprenda aos poucos a usar os conceitos apresentados em cada nível. Para isso, os primeiros problemas de cada nível são mais simples que os últimos. Essa estratégia faz parte do conceito de retórica procedural e é utilizada para fundamentar melhor o conhecimento lógico do jogador e aumentar a sua capacidade de resolver os problemas de salas seguintes.

F. Blocos

Blocos são a representação dos principais comandos de programação básica dentro do jogo. Eles são utilizados para resolver problemas no terminal, por meio da criação de códigos de programação formados a partir da ligação entre vários blocos. Todos os blocos possuem um ponto de conexão de entrada, por onde outros blocos se conectarão a ele e entre um e três pontos de conexão de saída, utilizados para se conectar aos outros blocos do código.

Cada bloco é a representação de um comando. A Tabela II mostra todos os blocos, quais comandos eles representam e a partir de qual nível de dificuldade eles são necessários.

TABELA II
BLOCOS DISPONÍVEIS DENTRO DO JOGO

Bloco	Representação	Necessário em
Variável	Criação de uma variável	Todos os níveis
Leitura	Comando scanf	Todos os níveis
Imprime	Comando printf	Todos os níveis
Lógica ou aritmética	Realizar operações	Todos os níveis
Condição	Blocos if e else	A partir do nível 2
Laço definido	Bloco for	A partir do nível 3
Laço indefinido	bloco while	A partir do nível 3
Vetor	Criação de um vetor	A partir do nível 4
Matriz	Criação de uma matriz	A partir do nível 4

Podemos ver um exemplo de bloco na Fig. 6 que mostra um bloco condicional, ou, if-else. Esse bloco possui um texto

que exibe qual a operação que deverá ser resolvida por ele, sendo, neste caso, a seguinte operação de comparação: se a variável a é menor ou igual à variável b. O bloco possui também quatro pontos de conexão. Esses pontos servem para conectar o bloco a outros blocos, mantendo o fluxo do código. Na Fig. 6, por exemplo, podemos ver que outro bloco já está conectado ao bloco condicional pelo lado esquerdo, no ponto de conexão de entrada, o que significa que esse outro bloco será executado antes do bloco condicional. Os pontos de conexão acima e abaixo do bloco são as conexões dos desvios condicionais: caso a operação seja verdadeira, o bloco conectado ao ponto de baixo será executado; caso a operação seja falsa, o bloco acima será executado. E, por último, temos a conexão da direita, que é executada imediatamente após o fluxo condicional terminar, ou seja, quando todos os blocos em cadeia que estão conectados à conexão condicional forem executados.

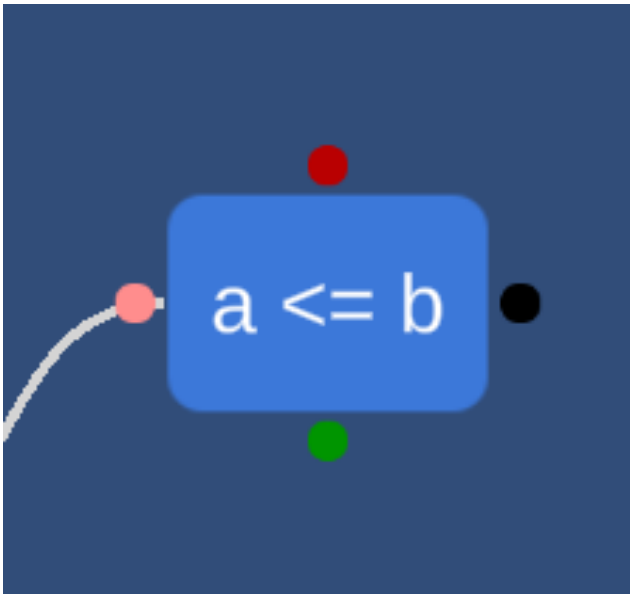

Fig. 6. Representação do bloco if-else.

Continuando o exemplo do bloco condicional, temos na Fig. 7 a janela de edição do bloco, onde a operação do bloco é construída pelo jogador. No lado esquerdo temos uma lista com todas as variáveis criadas acessíveis ao jogador para inseri-las na operação, e do lado direito há um teclado que lembra uma calculadora, onde é possível adicionar valores fixos e ter acesso a vários tipos de operações lógicas.

G. Histórico de versões

Ao longo do desenvolvimento o terminal teve uma versão apenas conceitual, vista na Fig. 8, e duas versões jogáveis. Essas versões do jogo, incluindo a conceitual, foram testadas e/ou analisadas por alunos do primeiro e segundo período do curso a fim de serem validadas. A partir do *feedback* dos alunos e a ajuda do *design* participativo, mudanças foram feitas para que o jogo se tornasse cada vez mais interessante para o seu público-alvo.

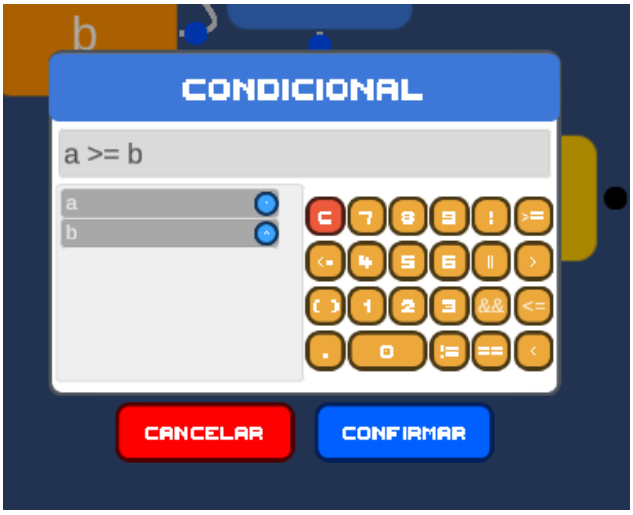

Fig. 7. Janela de edição do bloco if-else.

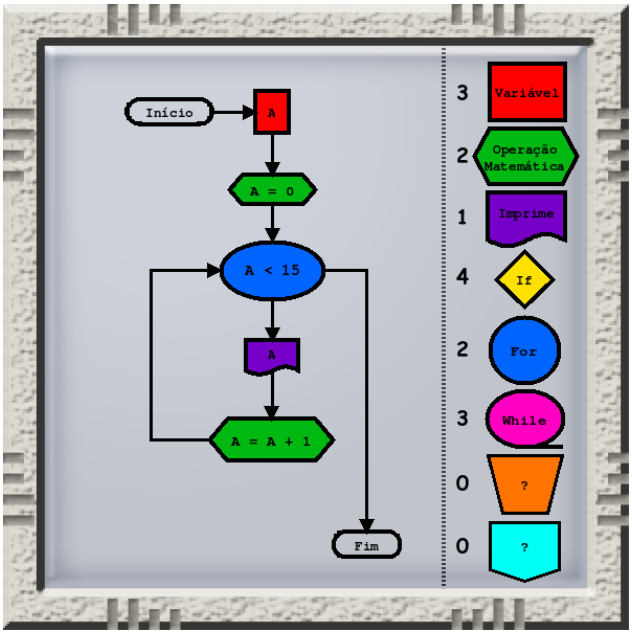

Fig. 8. Primeiro conceito do terminal.

A primeira versão jogável do terminal, vista na Fig. 9, utilizava um *asset* para Unity chamado Roslyn C#[3]. O Roslyn C# permite a compilação em tempo de execução de *scripts* em C#, facilitando a execução de códigos construídos pelo próprio jogador por meio dos blocos. Além disso, o Roslyn C# também inclui verificação de segurança do código que permite que sejam especificadas restrições de segurança que o código deve atender, incluindo namespaces e tipos ilegais.

Porém, essa abordagem, vista na Fig. 10, tinha alguns problemas. O primeiro deles é que os códigos compilados não tinham interação nenhuma com o jogador. Até mesmo as en-

[3]Roslyn C# - https://assetstore.unity.com/packages/tools/integration/roslyn-c-runtime-compiler-142753

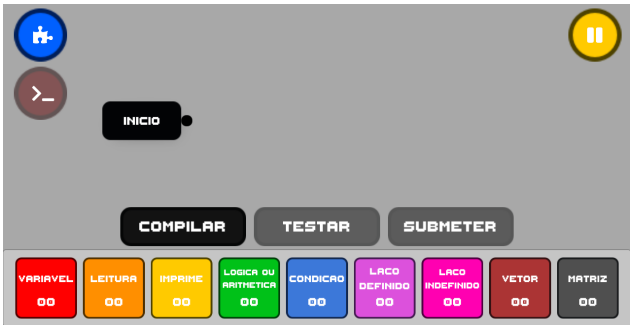

Fig. 9. Terminal em sua primeira versão.

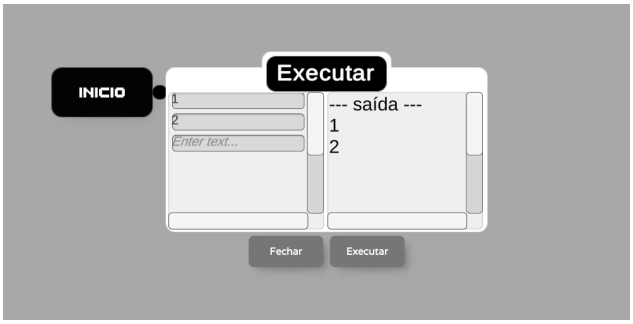

Fig. 11. Janela de execução de um código na primeira versão.

tradas do código tinham que ser pré-carregadas na compilação, causando incômodo a alguns jogadores que testaram essa versão. Esta abordagem também apresentava dificuldade na utilização de vários tipos de dados. Por conta de limitações da ferramenta, apenas o tipo *int* estava disponível no jogo. E por fim, algumas janelas eram confusas demais, como a janela de execução de códigos vista na Fig. 11 onde o jogador deveria inserir todas as entradas antes de executar o código para que essas entradas fossem inseridas no código durante a compilação.

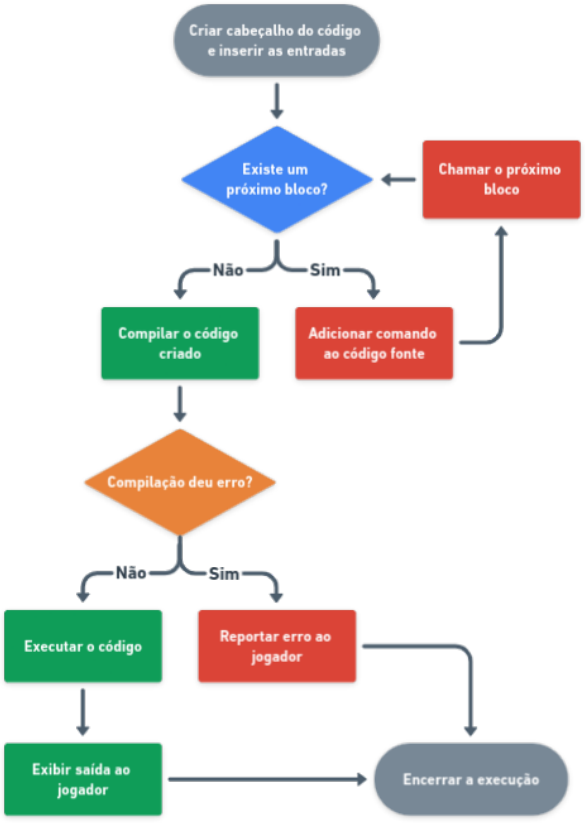

Fig. 10. Fluxo de execução de um código na primeira versão.

Já a segunda versão do terminal funciona sendo totalmente interpretada pelo jogo, descartando totalmente a compilação e o uso do Roslyn C#. Nesta versão os códigos são representados por uma espécie de grafo direcional, onde cada bloco representa um nó desse grafo e suas conexões são as arestas. Sempre que um bloco é executado, ele chama a execução do bloco seguinte, e essa interação se repete até que o código termine, seja encerrado ou encontre um erro.

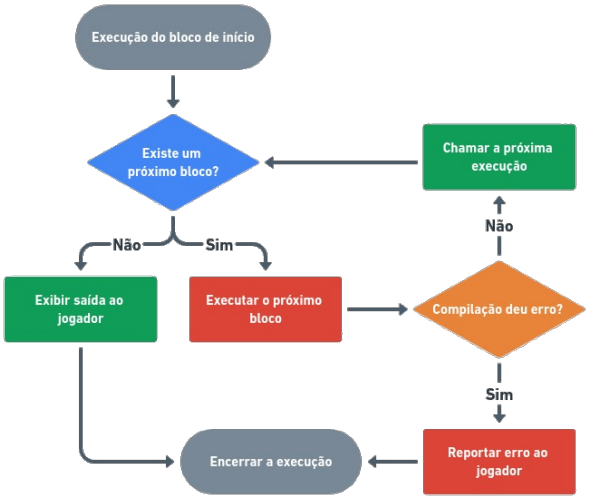

Fig. 12. Fluxo de execução de um código na versão atual.

Ao utilizar a segunda abordagem, vista na Fig. 12, foi possível adicionar ao jogo os tipos *float*, *boolean* e *string*, fazendo com que os problemas pudessem ser mais diversos. Além dos novos tipos, foi possível adicionar um formato de execução iterativa, mostrado na Fig. 13, onde o jogador consegue observar o seu código sendo executado e inserir entradas para teste em tempo real, facilitando a busca por erros de lógica e aproximando o jogador de uma situação real.

VI. Resultados e Discussão

Nesta Seção são apresentados os resultados dos questionários disponibilizados ao longo do desenvolvimento deste trabalho. Inicialmente é apresentado um questionário de pré-desenvolvimento realizado em 2019, onde é possível ver re-

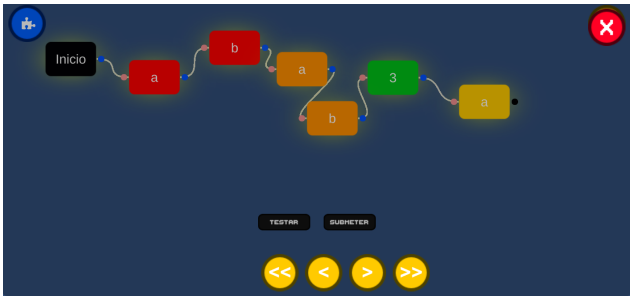

Fig. 13. Código executando na versão atual.

sultados que apontam uma boa aceitação por parte do público-alvo. Em seguida tem-se um questionário referente à versão beta do jogo, que serviu como um métrica para avaliarmos se o jogo estava no caminho certo e realizar decisões importantes sobre a sua experiência de jogo. E por último, apresentamos os dois questionários finais que trazem resultados que mostram uma evolução do jogo para uma versão estável.

A. *Questionário pré-desenvolvimento (2019)*

Esse questionário teve um total de 14 respostas. Como principais resultados podemos observar na Fig. 14, Fig. 15 e na Fig. 16 que os respondentes consideraram em sua maioria que tem um conhecimento de médio para baixo em programação, mas que tem interesse no jogo e que acreditam que ele pode ajudar no aprendizado.

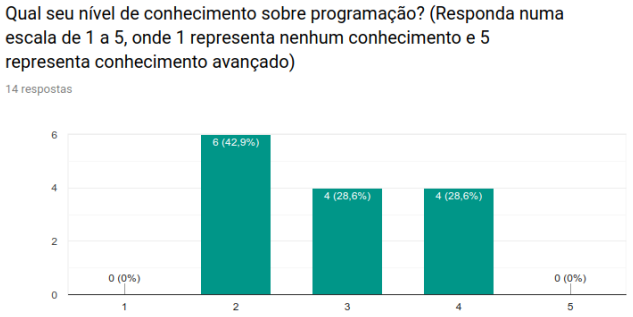

Fig. 14. Nível de conhecimento sobre programação.

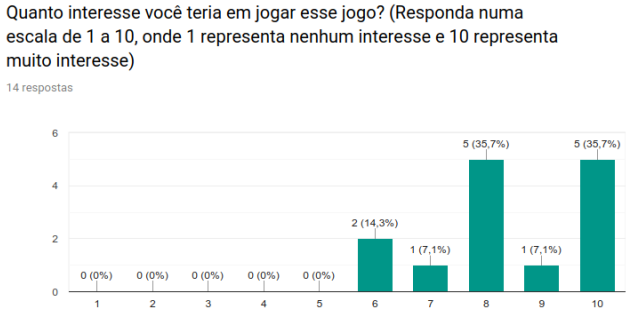

Fig. 15. Interesse em jogar o Maze Code.

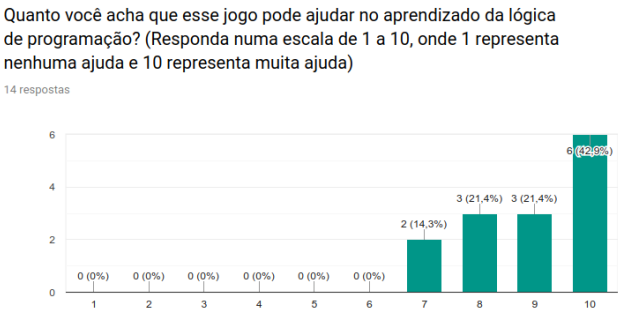

Fig. 16. Contribuição do Maze Code no aprendizado de programação.

B. *Questionário desenvolvimento (versão BETA) (2020)*

Contendo apenas uma questão aberta com foco na experiência de jogo até o momento, obtivemos 11 respostas que foram fundamentais para correção de *bugs* críticos e melhorias para o lançamento da versão estável.

Como exemplo de respostas que nos ajudaram a corrigir bugs e/ou ajudaram no desenvolvimento da versão estável, temos: Senti falta de alguma indicação de como era pra usar o compilador, e fiquei um tempo construindo meu programa e quando cliquei em compilar ele sumiu; Gostei muito da idéia, apesar de apresentar alguns bugs como o personagem desaparecer após o desafio ou o comando início não aparecer algumas vezes. O jogo em si foi bem desenvolvido e a mecânica é bem simples; Consegui clicar em 'continuar' sem ter dado Novo Jogo; Interface interessante. Difícil manusear a ligação entre os blocos de programação. Código não foi aceito apesar de estar correto; A única coisa negativa pra mim foi o fato da criação dos códigos não ser muito atrativa, apesar de ela ter uma ideia boa e ser simples, parece que falta algo.

C. *Questionários pós-desenvolvimento (2020 e 2021)*

O primeiro questionário, aplicado em 2020 após o desenvolvimento da primeira versão do jogo, obteve resposta de 8 alunos e o resultado pode ser visto na Tabela III.

Já o segundo questionário foi aplicado em 2021, após o desenvolvimento da segunda versão do jogo, com melhorias em relação à jogabilidade, e obteve resposta de 11 alunos e seu resultado pode ser visto na Tabela IV.

Realizando uma análise mais detalhada nos resultados obtidos com esses dois questionários, é possível notar que há uma melhora considerável em relação à experiência do jogo em geral, tendo respostas mais concentradas nos níveis mais altos. Também há uma perceptível evolução nas respostas quando se trata da interface de interação do jogo, pois foram realizadas diversas mudanças considerando o design participativo para obtermos uma interface mais interativa e fácil de se usar, e de acordo com o público-alvo. Seguindo as respostas do primeiro questionário, foram realizadas melhorias na interface a fim de ficar mais claro o que o jogador deve fazer durante o jogo, assim obtivemos uma melhora nas respostas nesse quesito. Outro impacto positivo foi em relação às mensagens e avisos no jogo, pois na versão de 2020 percebemos com o

TABELA III
RESPOSTAS DO QUESTIONÁRIO DA PRIMEIRA VERSÃO ESTÁVEL DO JOGO

pergunta	1	2	3	4	5
A utilização do jogo em geral foi uma experiência satisfatória?	0%	25%	12,5%	37,5%	25%
A interface de interação do jogo pode ser facilmente compreendida?	0%	25%	37,5%	25%	12,5%
As informações contidas na interface do jogo são suficientes para sua utilização?	12,5%	12,5%	12,5%	37,5%	25%
Durante a utilização do jogo as mensagens e avisos são suficientes para compreensão das tarefas e desafios?	0%	37,5%	25%	25%	12,5%
As funcionalidades necessárias para a realização das tarefas podem ser encontradas com facilidade?	12,5%	0%	25%	12,5%	50%
Os desafios apresentados estão de acordo com as dificuldades propostas?	0%	0%	12,5%	12,5%	75%
O jogo contribui para um melhor entendimento de lógica de programação?	0%	12,5%	12,5%	12,5%	62,5%

TABELA IV
RESPOSTAS DO QUESTIONÁRIO DA SEGUNDA VERSÃO ESTÁVEL DO JOGO

pergunta	1	2	3	4	5
A utilização do jogo em geral foi uma experiência satisfatória?	0%	0%	18,1%	36,4%	45,5%
A interface de interação do jogo pode ser facilmente compreendida?	0%	18,2%	9,1%	45,5%	27,3%
As informações contidas na interface do jogo são suficientes para sua utilização?	9,1%	0%	27,3%	9,1%	54,5%
Durante a utilização do jogo as mensagens e avisos são suficientes para compreensão das tarefas e desafios?	0%	9,1%	18,2%	36,4%	36,4%
As funcionalidades necessárias para a realização das tarefas podem ser encontradas com facilidade?	0%	0%	27,3%	18,2%	54,5%
Os desafios apresentados estão de acordo com as dificuldades propostas?	0%	0%	0%	27,3%	72,7%
O jogo contribui para um melhor entendimento de lógica de programação?	0%	0%	18,2%	18,2%	63,6%

questionário que faltavam avisos e mensagens que tornassem o objetivo do jogador mais claro. Dessa forma, realizamos melhorias e foram adicionados personagens que explicassem um pouco para o jogador o objetivo dele, o que fez com que os resultados no segundo questionário fossem mais positivos. Foi notado um pequeno avanço quando se trata das funcionalidades disponíveis para realização das tarefas, e também quando se trata do nível dos desafios apresentados no jogo. Por último, há uma pequena melhoria em relação à contribuição do jogo para um melhor entendimento de lógica de programação.

VII. CONCLUSÃO

O trabalho é fundamentado na construção de um jogo que tem como objetivo aumentar o engajamento e o aprendizado de conceitos relacionados à lógica de programação a partir da resolução de problemas em uma interface de programação por blocos, tendo como base para a sua criação e desenvolvimento, a retórica procedural e o design participativo.

A criação e o desenvolvimento do jogo, com o auxílio do design participativo, mesmo em um contexto remoto devido à pandemia atual, teve como objetivo aproximar seu público-alvo, que são os alunos do curso, de todo o processo de produção do jogo, por meio de vários *feedbacks* ao longo do desenvolvimento.

Quanto aos resultados, foram avaliados quesitos como a usabilidade das interfaces, intuitividade dos problemas apresentados, a jogabilidade e o aprendizado sobre o assunto proposto em cada versão. Podemos perceber um alto número de respostas positivas, principalmente sobre a versão atual, demonstrando que o uso do design participativo foi importante para tais resultados. Além disso, a retórica procedural se mostra eficaz no contexto apresentado, visto que não houve respostas negativas sobre o entendimento do assunto proposto.

Em trabalhos futuros pretende-se criar mais problemas e salas, expandindo o mapa do jogo, a adição de outros conceitos vistos em disciplinas introdutórias à programação, como funções e estruturas, e aplicar o jogo em alguns alunos de uma turma do primeiro período do curso. É pretendido aplicar o jogo em apenas um grupo de alunos aleatório para que exista um grupo de controle e que seja possível comparar os dados entre os alunos que jogaram e os que não jogaram o Maze Code durante a disciplina.

AGRADECIMENTOS

Por fim, gostaríamos de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio dado a este trabalho.

REFERÊNCIAS

[1] Y. Bosse and M. Gerosa, "Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar," *Anais do Workshop sobre Educação em Computação (WEI)*, 2015. [Online]. Available: https://sol.sbc.org.br/index.php/wei/article/view/10259

[2] M. De Aguilera and A. Mendiz, "Video games and education:(education in the face of a "parallel school")," *Computers in Entertainment (CIE)*, vol. 1, no. 1, p. 1, 2003.

[3] J. McGonigal and E. Rieche, *A realidade em jogo: Por que os games nos tornam melhor e como eles podem mudar o mundo*. Editora Best Seller, 2012.

[4] E. S. Association, "Essential facts about the computer and video game industry," 2020, acessado em 24/06/2021. [Online]. Available: https://www.theesa.com/resource/2020-essential-facts/

[5] M. N. Nascimento, M. S. Nery, and V. Silva, "Desenvolvimento de jogos digitais e sua utilização na educação juvenil: Um estudo de caso real em um projeto governamental," *SBC–Proceedings of SBGames*, 2013.

[6] C. A. F. da Silva, L. D. da Silva, and J. C. D. Martins, "Aplicação do the huxley no ensino de programação para alunos do curso técnico em informática para internet," *SBC–Proceedings of SBGames*, 2018.

[7] I. M. M. Santos, J. S. M. A. H. Kavalerski, and T. J. G. de Souza, "As aventuras espaciais de cody: protótipo de jogo para auxiliar no ensino de lógica de programação," *SBC–Proceedings of SBGames*, 2018.

[8] R. V. da Rocha, I. I. Bittencourt, and S. Isotani, "Análise, projeto, desenvolvimento e avaliação de jogos sérios e afins: uma revisão de desafios e oportunidades," *Anais do Xxvi Simpósio Brasileiro de Informática na Educação (sbie 2015)*, 2015.

[9] A. M. A. Price and S. S. Toscani, *Implementação de Linguagens de Programação: Compiladores*. Porto Alegra: Sagra Luzzatto, 2001.

[10] A. Aho, R. Sethi, and S. Lam, *Compiladores: princípios, técnicas e ferramentas*. PRENTICE HALL BRASIL, 2008. [Online]. Available: https://books.google.com.br/books?id=hahXPgAACAAJ

[11] R. d. Santiago and R. L. S. Dazzi, "Interpretador de portugol," in *Artigo submetido e aprovado para o IV Congresso Brasileiro de Computação. Universidade do Vale do Itajaí–UNIVALI. Itajaí*, vol. 8, 2004.

[12] I. Bogost, "The rhetoric of video games," *The ecology of games: Connecting youth, games, and learning*, pp. 117–140, 2008.

[13] M. J. Muller and S. Kuhn, "Participatory design," *Commun. ACM*, vol. 36, no. 6, p. 24–28, Jun. 1993. [Online]. Available: https://doi.org/10.1145/153571.255960

[14] R. L. M. P. N. Mustaro, "Elementos imersivos e de narrativa como fatores motivacionais em serious games," *SBC–Proceedings of SBGames*, 2011.

[15] R. Gomes, "The design of narrative as an immersive simulation." in *DiGRA Conference*, 2005.

[16] J. Macena, F. Pires, and M. Pessoa, "Operaçao lovelace: uma abordagem ludica para introduçao de aprendizagem em algoritmos," *SBC–Proceedings of SBGames*, 2020.

[17] I. M. M. Santos, J. Kavalerski, and T. J. G. de Souza, "As aventuras espaciais de cody: protótipo de jogo para auxiliar no ensino de lógica de programação," *XVII SBGames. Scaico, PD, de Lima, AA, Azevedo, S., da Silva, JBB, Raposo, EH, Alencar, Y., & Scaico, A*, 2013.

[18] A. A. de Souza, M. L. de Oliveira, A. K. Tenório, H. d. O. Renato, and A. N. Rodrigues12, "Mundo de euclides: Aplicabilidade de um jogo para o ensino da geometria euclidiana," *SBC–Proceedings of SBGames*, 2014.

[19] K. A. de Godoi and S. Padovani, "Instrumentos avaliativos de software educativo: uma investigação de sua utilização por professores," *Estudos em Design*, vol. 19, no. 1, 2011.