Explorando a lógica de programação através de um jogo com auxílio da realidade aumentada

Bruno Junkes, Dalton Solano dos Reis, Luciana Pereira de Araújo Kohler, Mauricio Capobianco Lopes, Miguel Wisintainer, Lucas Hong Lae Son

[1] Laboratório de Desenvolvimento e Transferência de Tecnologias (LDTT)
Departamento de Sistemas e Computação (DSC)
Universidade Regional de Blumenau (FURB)
Blumenau – SC – Brasil.

{brjunkes, dalton, lpa, mclopes, maw, lson}@furb.br

Abstract. *This paper presents a game for Android and iOS systems, with the aim to support to learn programming logic, with focus on first semesters of Computer Science course. The game was developed using the Unity engine together with the AR Foundation framework, which allows working with the concept of Augmented Reality (AR). Through blocks with code commands, the player should create an algorithm indicating which path the characters should follow to reach their destination. Tests were carried out with a public, in order to evaluate its receptivity and effectiveness. The goals were achieved, as players were able to interact with the AR and understand how each block works, thus completing a large part of the proposed levels.*

Resumo. *Este artigo apresenta um jogo para sistemas Android e iOS, com o intuito de auxiliar no aprendizado da lógica de programação, tendo maior foco nas fases iniciais de ensino dos cursos de Computação. O jogo foi desenvolvido utilizando o motor de jogos Unity juntamente ao framework AR Foundation, que permite trabalhar com o conceito de Realidade Aumentada (RA). Através de blocos com comandos, o jogador deve montar um algoritmo indicando qual caminho os personagens devem seguir para chegar ao seu destino. Foram realizados testes com um público, com o intuito de avaliar sua receptividade e eficácia. Os objetivos foram atingidos, visto que os jogadores conseguiram interagir com a RA e compreender o funcionamento de cada bloco, concluindo assim boa parte dos níveis propostos.*

1. Introdução

É notável que nos últimos anos, uma das áreas que mais tem crescido é a da computação e com isso, exigindo cada vez mais profissionais qualificados para atuar em suas diversas subáreas. Contudo, sabe-se que esta não é uma área fácil, trazendo uma alta complexidade consigo e fazendo com que muitos tenham dificuldades em absorver conteúdos relacionados.

Devido a estas dificuldades, acaba-se culminando na desistência por grande parte dos estudantes de computação, muitas vezes resultante de metodologias defasadas de ensino utilizados por seus professores, complexidade da lógica de programação e falta de conhecimento de conceitos básicos [Souza et al. 2016]. [Morais et al. 2020] (p.3) também apontam que:

DOI: 10.5753/wpci.2024.245486

> cada aluno possui suas dificuldades individuais, seu ritmo de aprendizagem, seus interesses e motivações, de forma que os professores precisam identificar as características de seus alunos e as dificuldades por eles enfrentadas para que consigam dar o suporte devido.

Tendo estas dificuldades em mente, faz-se necessária a criação ou adaptação de métodos utilizados para lecionar as matérias que envolvam a área da computação. Dentro da computação, uma área que vem crescendo é a Realidade Aumentada (RA). A RA é a combinação de elementos do mundo real e do virtual em tempo real. [Aragão et al. 2023] realizaram um Mapeamento Sistemático da Literatura (MSL) em que destacam que a RA junto com a Realidade Virtual (RV) são utilizadas em aplicações educacionais para o ensino da programação e que seu uso demonstra um potencial para melhorar a qualidade do ensino. Nesse contexto, usar a tecnologia da RA é favorável, pois cria um ambiente descontraído que chama a atenção e desperta o interesse das pessoas.

Com base no exposto, este trabalho apresenta um jogo que poderá ser usado como ponto de partida para o estudo da lógica de programação nas fases iniciais de ensino nos cursos de graduação em computação. Assim, o objetivo principal deste trabalho é propor um jogo para ensino de lógica de programação, utilizando RA.

O artigo segue estruturado da seguinte forma. A seção 2 discute três trabalhos relacionados ao apresentado neste artigo, destacando os diferenciais do trabalho em questão. A seção 3 apresenta o desenvolvimento do jogo, trazendo uma visão geral sobre o mesmo, as especificações e sua implementação. A seção 4 discute sobre os testes de funcionalidade e com o usuário realizados. Por fim, a seção 5 discute sobre as considerações finais do trabalho.

2. Trabalhos correlatos

A seguir são apresentados três trabalhos correlatos ao apresentado neste artigo. O primeiro é o trabalho de [Saraiva 2022] que tem como objetivo ajudar a desenvolver os quatro pilares do PC: reconhecimento de padrões,decomposição, algoritmos e abstração. A principal funcionalidade do trabalho é a utilização de cartões físicos que representam ações que um determinado personagem pode executar, para construir um caminho lógico na tentativa de guiar todos os personagens ao seu destino. Para o desenvolvimento do jogo foi utilizada a Unity, configurada para sistemas Android. Já para criar a visualização em RA utilizou-se a plataforma Vuforia, que realiza o processamento de imagens gerando marcadores virtuais. Outra ferramenta utilizada foi o Blender, que permite criar animações, efeitos e modelos 3D. [Saraiva 2022] conclui que o objetivo de desenvolver um jogo em RA que colaborasse com o estudo do PC foi alcançado e que a partir dos estudos, pesquisas, apresentação e *feedbacks* coletados é realmente possível treinar habilidades de PC através de jogos.

O segundo trabalho correlato é o Robo Kyle desenvolvido por [de Mello and Antoniazzi 2021] que tem o objetivo de criar uma metodologia de ensino que proporcione um meio mais divertido e desafiador para estudantes sobre lógica e programação. Outro objetivo é aumentar o interesse nas área da informática, fornecendo um acesso facilitado ao estudo desses assuntos. A ferramenta desenvolvida por eles apresenta conceitos relacionados a lógica e algoritmos no início de cada fase e *joysticks* para movimentação livre do personagem pelos cenários afim de resolvê-los.

Para o desenvolvimento do jogo foi utilizada a Unity, configurada para sistemas Android. Utilizou-se a plataforma Asset Store para a coleta de modelos 3D para os cenários e personagem. Por fim, foi utilizado também a plataforma Vuforia para a criação dos marcadores virtuais. Segundo [de Mello and Antoniazzi 2021], após a conclusão do desenvolvimento do jogo, planeja-se apresentá-lo em escolas com o intuito de aplicar questionários que poderão avaliar a eficácia dele em relação ao aprendizado dos conceitos envolvidos na programação.

O terceiro trabalho correlato desenvolvido por [KUNTZ 2020] tem como objetivo criar um módulo de IUT para um jogo com o intuito de treinar o Pensamento Computacional. Para isso, foram realizadas detecções do contorno de imagens para traduzi-las em comandos para o personagem principal. Para o desenvolvimento do jogo foi utilizada a Unity, juntamente ao pacote OpenCV For Unity que oferece métodos para detecção de imagem. Para criação das peças que serão interpretadas pelo jogo, foi utilizado o software de edição de imagens Photoshop. Segundo o autor, embora ainda existam limitações, como a influência da luz ambiente, sombras, posição e ângulo das peças, o trabalho atingiu seus objetivos, apresentando uma aplicação Android que reconhece e executa comandos com base nas peças criadas pelos usuários.

A partir dos trabalhos apresentados, o trabalho em questão denominado ProgramAR se mostra relevante como uma ferramenta de aprendizagem de lógica. O jogo busca ensinar utilizando uma maior interatividade com os cenários, dando aos jogadores uma maior liberdade ao permitir fixar o jogo em qualquer lugar e possibilitando andar ao seu redor para visualizar possíveis caminhos, armadilhas ou obstáculos, proporcionando assim uma maior imersão.

3. Descrição do Jogo

Na presente seção serão descritos os detalhes de especificação e implementação do jogo desenvolvido nomeado como ProgramAR, a qual está dividida em três subseções. Na subseção 3.1 é apresentada uma visão geral dos elementos presentes no jogo. Na subseção 3.2 tem-se a especificação. Na subseção 3.3 são abordados detalhes das implementações das funcionalidades presentes no jogo.

3.1. Visão geral

O jogo ProgramAR traz ao jogador desafios que envolvem criar uma sequência lógica de comandos para construir um caminho, essa sequência é conhecida como algoritmo que o jogo introduz como um mapa. Esse mapa tem como objetivo guiar os personagens, dois astronautas perdidos em outro planeta, de volta para o seu foguete.

Para a construção dos mapas cada astronauta tem acesso a um painel próprio, em que o jogador pode adicionar diversos blocos com comandos, cada um representando uma ação que pode ser executada pelos astronautas, são elas: andar um bloco a frente, rotacionar para a direita ou esquerda, interagir com os objetos que estiverem a frente e esperar, que mantém um astronauta parado até que o outro finalize sua próxima ação. Além disso, o jogo trabalha com a introdução ao conceito de funções de programação, trazendo níveis que exigem que o jogador utilize sequências de comandos em um painel específico para funções, que possam ser repetidas várias vezes em pontos distintos de seu algoritmo.

Durante o desenvolvimento do mapa, o jogador também deve se ater aos desafios propostos pelos cenários. Desafios esses que podem envolver apenas um astronauta específico, como coletar uma chave para abrir uma porta que está bloqueando seu caminho, ou envolver a colaboração dos dois astronautas, como um deles puxar uma alavanca para abrir a porta que está bloqueando o outro ou até mesmo manter um botão pressionado para desativar espinhos que matariam o astronauta que passar por cima.

Como um meio de ajudar o jogador quando não estiver conseguindo construir uma lógica que consiga levar os astronautas até o foguete, o jogo também conta com uma tela de *logs* de erro. Essa tela informa ao jogador as ações que os astronautas tentaram executar e não conseguiram juntamente a descrição dos possíveis motivos, assim dando uma ideia de quais comandos o jogador deve alterar para consertar sua lógica.

O jogo divide todas estas funcionalidades em duas páginas de níveis, trazendo na primeira uma introdução às ações básicas já citadas e a segunda o conceito de funções, juntamente a uma breve explicação sobre *loops* infinitos, estruturas de repetição que não possuem um fim definido. Cada conceito é explicado detalhadamente a partir de tutoriais, em que um personagem não jogável chamado Iara conversa com o jogador explicando o objetivo do jogo, como utilizar os blocos com comandos e como cada ação é executada pelos astronautas. A Figura 1 demonstra uma visão do cenário projetada em um ambiente real (sendo que no fundo aparece um chão real), junto a tela de programação à direita e a tela de *logs* no canto inferior esquerdo.

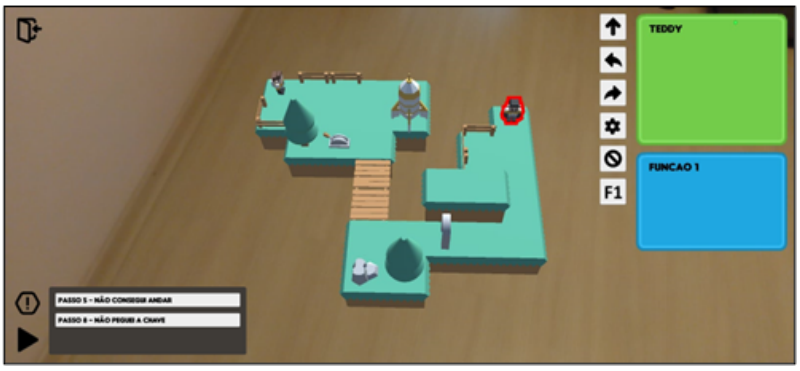

Figura 1. Cenário do jogo

3.2. Especificação

Esta seção apresenta a arquitetura do jogo desenvolvido. A Tabela 1 apresenta os Requisitos Funcionais (RF) desenvolvidos. Já a Figura 2 apresenta o fluxograma dos processos do jogo. Com o início do fluxograma é solicitado ao usuário a permissão de uso da câmera do aparelho, necessária para que a realidade aumentada possa ser utilizada, caso contrário, o jogo não poderá montar os cenários e assim, nenhuma das interações ficará disponível para o jogador. Com a permissão concedida, o jogador será redirecionado para o menu inicial, que lhe permitirá sair do jogo ou iniciá-lo, sendo redirecionado para a tela de seleção de níveis, que por sua vez permitirá voltar para o menu inicial ou selecionar um nível. Quanto à seleção de níveis, existem atualmente duas páginas, a primeira contendo níveis que apresentaram os comandos básicos do jogo (andar, girar, interagir e esperar) e a segunda trazendo o conceito de função.

Tabela 1. Requisitos funcionais

RF01: permitir que o jogador navegue entre os menus do jogo
RF02: permitir a locomoção em volta do cenário que compõe os níveis com o intuito de aumentar a interatividade com o jogo
RF03: permitir a interação com as entidades dos níveis a partir do dispositivo móvel
RF04: permitir que seja possível montar algoritmos utilizando os blocos com comandos com o intuito de desenvolver a lógica de programação
RF05: permitir que seja possível visualizar os blocos que compõem o código de cada entidade
RF06: permitir que seja possível visualizar os logs de erro que impediram o sucesso do nível
RF07: permitir que os blocos do código das entidades sejam reprogramados
RF08: permitir executar os códigos programados com os blocos, para visualizar em tempo real a lógica desenvolvida

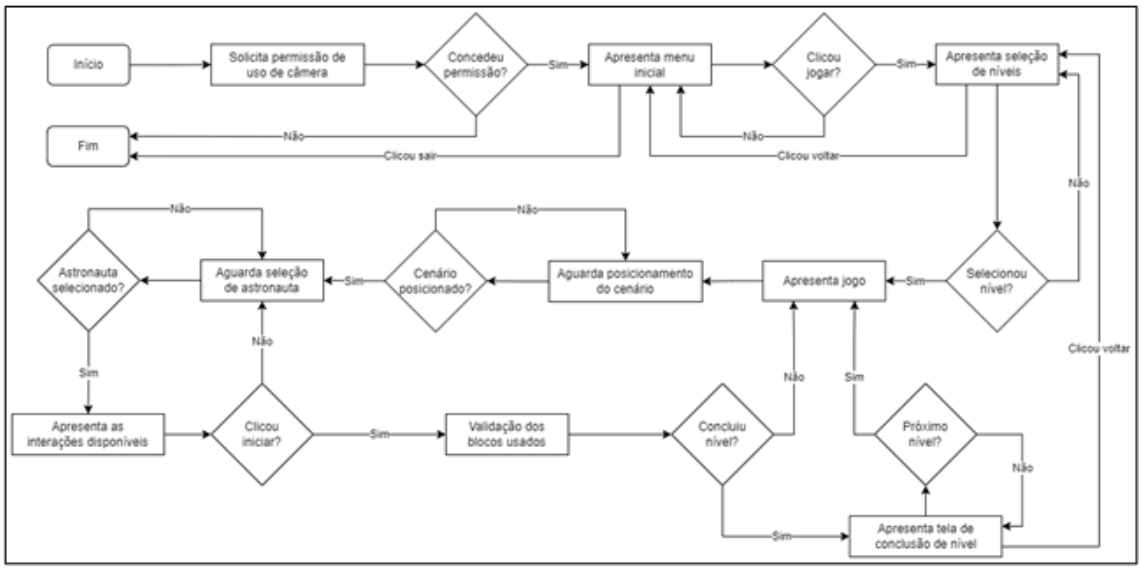

Figura 2. Fluxograma da tela do jogo

Ao selecionar um nível, a tela principal do jogo é aberta e apresentará o cenário proposto ao jogador, que deverá mover o aparelho e clicar na tela para selecionar o local que deseja fixar este cenário. Tendo o posicionado, será possível selecionar um dos astronautas apontando a câmera para eles e clicando na tela, o que abrirá a tela de programação e apresentará os blocos com comandos disponíveis para o nível. Após montar a lógica, o jogador poderá executar os comandos a partir do botão iniciar, que então irá traduzir os blocos com comandos em funções internas e validará se ambos os astronautas chegaram no destino. Em caso positivo, será apresentada a tela de conclusão de níveis, que permitirá voltar a tela de seleção de níveis ou avançar para o próximo caso exista. Durante a execução dos comandos, o jogador poderá visualizar os astronautas andando pelo cenário, para que em casos de falha, ele possa entender melhor onde errou.

Outra interação que é disponível ao clicar em um astronauta, caso o jogador já tenha executado os comandos pelo menos uma vez e não tenha concluído o nível, é o

botão de *logs*. Assim que ele é pressionado será aberta uma tela de *log* de erros com informações dos comandos que falharam ao serem executados pelos astronautas, visando auxiliar o jogador a descobrir por que sua lógica falhou e o nível não ter sido concluído.

O jogo foi desenvolvido para os Sistemas Operacionais iOS versão 16.6 ou superior e Android 12 ou superior. Ainda, o *smartphone* deve conter ou ser compatível com o ARKit ou ARCore.

3.3. Implementação

Para a implementação do jogo foi utilizado o motor de jogos Unity na versão 2021.3.11f1 com sua linguagem de programação padrão C#. Para processar a RA foi utilizado o *framework* AR Foundation na versão 5.1. O desenvolvimento foi dividido em três partes, sendo elas a implementação dos blocos com comandos e *logs* de erro, validação da lógica desenvolvida e a visualização dos cenários com a RA.

Para controlar os blocos selecionados pelo jogador e da lista de *logs* de erro para cada astronauta, foi criada uma classe chamada *Algorithm*. Com relação aos blocos, caso o jogador selecione uma das opções de blocos disponíveis (Figura 3 (a)) será adicionado um novo identificador dentro de uma lista de controle nesta classe, ao mesmo tempo que se pressionado um bloco da lista (Figura 3 (b)), o identificador correspondente será removido da lista de controle. Para a lista de *logs* de erro, os itens são adicionados conforme os comandos dos blocos são executados e não conseguem concluir a ação correspondente. Para cada caso, é adicionada uma mensagem específica com o correspondente passo que falhou.

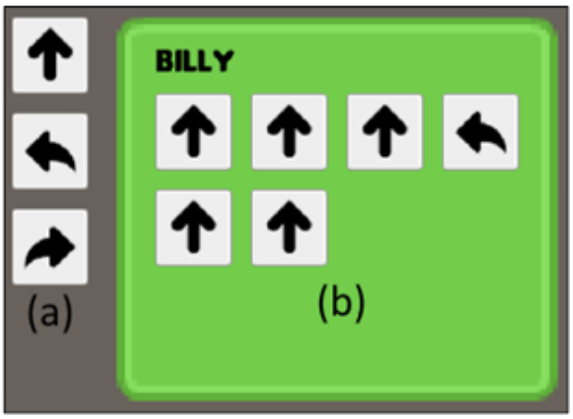

Figura 3. Tela de programação de um astronauta

4. Resultados

Esta seção apresenta os testes de funcionalidade realizados, bem como os testes com o usuário.

4.1. Testes de funcionalidades

Para realizar todos os testes da aplicação, visando seu funcionamento em ambos os sistemas operacionais, utilizou-se os *smartphones* iPhone 13 Pro com o iOS 16.6 e Samsung A52s com Android 12, mas qualquer versão ou modelo de dispositivos móveis que sejam superior e contenham o ARKit ou ARCore, devem suportar a aplicação desenvolvida.

Para realizar o desenvolvimento e os testes iniciais, utilizou-se o sistema operacional Windows 10 com uma configuração de 16 GB de RAM, com processador Intel Core i5 3.0 Ghz, com placa gráfica NVIDIA GeForce GTX 1660 e um SSD com velocidade de leitura de 545 MB/s.

Estes testes objetivaram encontrar o máximo de erros possível antes do jogo ser disponibilizado para outros usuários jogarem. Cada ação foi testada utilizando seu respectivo bloco com comando, de modo que para cada astronauta foram inseridos diversos deles em sequência, podendo assim visualizar se todas as animações e interações com o cenário foram executadas corretamente. Um exemplo das sequências criadas para ambos os astronautas em testes, é o apresentado na Figura 4. Para este teste foi utilizado o nível 4 da página de funções, que para ser resolvido necessita de todos os blocos com comandos atualmente disponíveis e o painel de função.

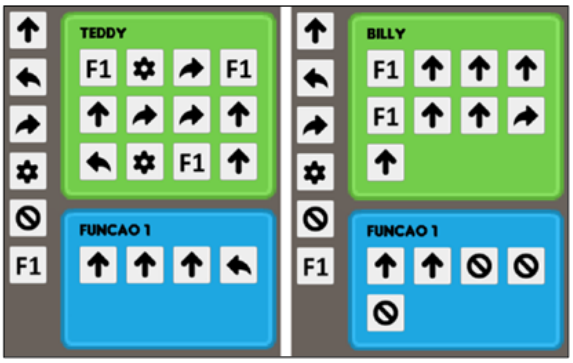

Figura 4. Exemplo de sequência correta de comandos utilizado em testes

Outros testes também incluíram os logs de erro, para os quais foram criadas lógicas incorretas intencionalmente, usando comandos de forma a gerar problemas de lógica. Como por exemplo, usar o comando de andar sendo que na frente do astronauta há um bloco bloqueando seu caminho, desta forma podendo testar se a tela de log de erros traria o erro corretamente. Quanto à rotina de validação da lógica criada pelo jogador, foram montadas diversas variações do algoritmo solução, todas percorrendo um caminho válido até o objetivo final, para garantir que todos os comandos estivessem sendo validados corretamente.

Com o intuito de encontrar possíveis problemas em aparelhos com configurações inferiores, foram realizados os mesmos testes com um MacBook mais antigo, neste momento sem o uso da RA, que executou o jogo com poucos quadros por segundo (*Frames Per Second* - FPS). Isso ocasionou uma lentidão no jogo e portanto, um mal funcionamento na ação de andar dos astronautas, visto que a animação de andar foi implementada com o uso de *delays*, que utilizam os FPSs como base e que caso sejam muito baixos, fazem com que o astronauta não consiga realizar um ciclo completo (andar de um bloco até outro), ou seja, a partir de uma ação de andar o astronauta dificilmente conseguirá chegar no objetivo por não andar a distância correta e ainda gerar erros de colisão incorretamente. Junto a isso, devido ao uso da RA exigir um processamento maior do aparelho, isso indicou também que em dispositivos móveis com configurações baixas, o jogo poderá apresentar problemas na ancoragem dos cenários.

O principal problema encontrado, acabou por forçar uma mudança na forma em

que a RA seria utilizada, sendo que a ideia inicial era o jogador selecionar a posição para ancoragem do cenário apenas no menu inicial e o jogo manter guardado esta informação até que ele fosse fechado. Contudo, o *ARRaycastManager* não estava conseguindo manter esta informação entre as telas do jogo, perdendo a referência a cada troca e a partir disso, o jogador terá de ancorar os cenários sempre que trocar de nível ou reiniciá-lo.

4.2. Testes com usuários

Com o objetivo de verificar a jogabilidade do jogo, foi elaborado e disponibilizado através do Google Forms um formulário a um grupo de 6 pessoas. O público possuía uma variação de idades entre 20 e 50 anos, na qual a grande maioria utiliza dispositivos móveis com frequência e já ouviram falar do conceito de RA.

Para avaliar o uso direto do jogo, foram feitas as seguintes perguntas: *(i)* Você conseguiu fixar os cenários com facilidade?; *(ii)* Você leu e achou os tutoriais intuitivos?; *(iii)* Você conseguiu entender como usar os blocos de ações?; *(iv)* Você conseguiu entender como funciona uma função?. A partir delas, notou-se que 1 jogador teve dificuldades em fixar os cenários, todos leram os tutoriais e 4 acharam que são intuitivos e 2 não acharam, todos eles conseguiram compreender como utilizar os blocos com comandos e como funcionam as funções.

Para avaliar a construção do jogo como um todo foram feitas as seguintes perguntas: *(i)* Como você classifica a usabilidade do ProgramAR em Geral?; *(ii)* Você acha que as interfaces do ProgramAR são amigáveis?; *(iii)* Você acha que o ProgramAR cumpriu seu objetivo de auxiliar no aprendizado da lógica?; *(iv)* Você achou intuitivo o uso da Realidade Aumentada no ProgramAR?; *(v)* Você achou interessante o uso de um jogo para ensinar lógica?. Como respostas a essa pergunta era uma escala de satisfação variando entre 1 e 5, na qual representam muito ruim a muito bom. No geral o jogo obteve uma avaliação boa, na qual a maioria das classificações ficaram entre 3 e 5. Para a usabilidade, 67% dos jogadores avaliaram como 4 e 33% avaliaram como 3, quanto às interfaces serem amigáveis, as avaliações ficaram entre 3 e 5 divididas com 33% dos jogadores cada. Sobre cumprir o objetivo de auxiliar no aprendizado da lógica, 84% concordaram que o jogo é uma opção para se trabalhar com este conceito. O uso da RA foi avaliada em 4 por 84% dos jogadores e em 3 por 16%, todos acharam interessante utilizar um jogo para abordagem da lógica de programação.

5. Considerações Finais

Com base nos testes realizados, o desenvolvimento do jogo ProgramAR proporcionou uma compreensão detalhada das complexidades e desafios inerentes à integração da Realidade Aumentada (RA) com a lógica de programação. Os testes extensivos com diferentes dispositivos revelaram questões cruciais de desempenho e usabilidade, destacando a importância da adaptação e otimização para uma variedade de configurações. Os desafios técnicos encontrados, como a adaptação da ancoragem dos cenários de RA, resultaram em mudanças significativas na estrutura do jogo, ilustrando a necessidade de flexibilidade e adaptação durante o processo de desenvolvimento.

Além disso, os testes com usuários proporcionaram uma melhoria sobre a receptividade e eficácia do jogo. A avaliação dos participantes revelou um entendimento positivo sobre a proposta de utilizar a RA como ferramenta para o aprendizado da lógica de

programação. Ainda que tenham sido identificadas algumas dificuldades na conclusão de todos os níveis e na ancoragem dos cenários, a percepção geral foi favorável, destacando a eficácia do jogo como um recurso complementar para o ensino e aprendizado desses conceitos.

As análises detalhadas dos testes proporcionaram uma maior visão com relação a aprimoramentos futuros. Por mais que os cenários possam ser renderizados sem a necessidade de marcadores, ainda existem limitações quanto ao ambiente, como por exemplo locais com muito reflexo, terrenos irregulares ou variações de luminosidade. Também existe a limitação de hardware que pode causar mau funcionamento caso o aparelho utilizado possua uma configuração muito básica, perderá facilmente o local em que o cenário foi ancorado e/ou executará o jogo com poucos FPS's, causando travamentos e incoerências na execução das ações programadas. Como trabalhos futuros serão realizados mais testes e implementadas correções e novas funcionalidades com base nos testes já efetuados.

Referências

Aragão, P., Avellar, G., and Barbosa, E. (2023). Ensino de programação e pensamento computacional utilizando realidade virtual, realidade aumentada e jogos: Um mapeamento sistemático da literatura. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 800–812, Porto Alegre, RS, Brasil. SBC.

de Mello, L. F. D. and Antoniazzi, R. L. (2021). Jogo com utilizaÇÃo de realidade aumentada voltado para o desenvolvimento lÓgico aplicado ao ensino fundamental e mÉdio. *REVISTA INTERDISCIPLINAR DE ENSINO, PESQUISA E EXTENSÃO*, 8(1):76–84.

KUNTZ, J. M. (2020). Interface de usuário tangível para trabalhar com o pensamento computacional no furbot. Master's thesis, Universidade Regional de Blumenau.

Morais, C., MENDES NETO, F., and Osório, A. (2020). Dificuldades e desafios do processo de aprendizagem de algoritmos e programação no ensino superior: uma revisão sistemática de literatura. *Research, Society and Development*, 9.

Saraiva, F. M. V. (2022). Building a game with augmented reality: for training computational thinking. Master's thesis, Universidade do Minho.

Souza, D., Batista, M., and Barbosa, E. (2016). Problemas e dificuldades no ensino de programação: Um mapeamento sistemático. *Revista Brasileira de Informática na Educação*, 24(1):39.