UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
COORDENAÇÃO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS RIBEIRO MADEIRA

CardSortRA: uma análise do uso de realidade aumentada no ensino de algoritmos

São Luís – MA.
2025.

Lucas Ribeiro Madeira

CardSortRA: uma análise do uso de realidade aumentada no ensino de algoritmos

Trabalho de conclusão de curso apresentado como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal do Maranhão.

Orientador: Prof. Dr. Carlos de Salles Soares Neto.

São Luís – MA.

2025.

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).

Diretoria Integrada de Bibliotecas/UFMA

Madeira, Lucas Ribeiro.
CardSortRA: uma análise do uso de realidade aumentada
no ensino de algoritmos / Lucas Ribeiro Madeira. - 2025.
49 p.
Orientador(a): Carlos de Salles Soares Neto.
Curso de Ciência da Computação, Universidade Federal do
Maranhão, Universidade Federal do Maranhão, 2025.
1. Realidade Aumentada. 2. Visualização de
Algoritmos. 3. Pensamento Computacional. 4. Informática
e Educação. I. Neto, Carlos de Salles Soares. II. Título

LUCAS RIBEIRO MADEIRA

CARDSORTRA: UMA ANÁLISE DO USO DE REALIDADE AUMENTADA NO ENSINO DE ALGORITMOS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: _____/_____/_____.

BANCA EXAMINADORA

_____

Orientador
Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

_____

Membro da banca
Prof. Dr Alexandre Cesar Muniz De Oliveira
Universidade Federal do Maranhão

_____

Membro da banca
Profa. Dra Li-Chang Shuen Cristina Silva Sousa
Universidade Federal do Maranhão

AGRADECIMENTOS

Primeiramente, expresso minha profunda gratidão a Deus por ter me permitido alcançar este momento. Sem Sua graça, não teria a capacidade nem a força para concluir este curso. Sou grato por Ele ter colocado em meu caminho amigos, familiares e professores que me apoiaram ao longo desta jornada. Toda a honra e toda a glória deste trabalho são d'Ele, pois nada disso seria possível sem Sua bênção.

Gostaria de agradecer também aos meus pais, Rubens e Cláudia, que sempre estiveram ao meu lado, incentivando-me a estudar, aprimorar-me e dar o meu melhor, tanto como pessoa quanto como profissional. Eles foram meus exemplos em todos os aspectos da vida. À minha irmã Larissa, que me convenceu a ingressar na área da computação; e à minha irmã Luana, que nestes últimos meses permaneceu acordada comigo até tarde, fazendo-me companhia e, muitas vezes, auxiliando na organização das ideias enquanto eu escrevia este TCC.

Agradeço, ainda, aos professores do departamento de Ciência da Computação da Universidade Federal do Maranhão pela oportunidade de aprender com vocês e por se disporem, em muitas ocasiões, não apenas a compartilhar seus conhecimentos, mas também suas experiências de vida. Essas contribuições foram, e têm sido, extremamente úteis academicamente, profissionalmente e, em muitos casos, como pessoa. Em especial, ao meu orientador Prof. Dr. Carlos de Salles Soares Neto, por ter sido um mentor desde o início do curso, por seus valiosos conselhos e por me proporcionar as primeiras oportunidades como programador e pesquisador. Agradeço, ainda, por sua paciência em não desistir de mim, mesmo quando eu me perdia em algum projeto mirabolante em que fazia questão de me envolver.

Meu agradecimento se estende aos meus amigos: Saulo, Breno Lucas, Fernando, Jonas, Beneilton, Felipe, Kleydson Beckman, Tadeu, por terem tornado esta caminhada mais leve, divertida e marcante desde o início. Principalmente, agradeço a João Leonardo e Priscila por terem permanecido comigo até os momentos finais desta jornada. Em especial à Priscila, que, mesmo sendo mãe do recém-nascido Carlos André, conseguiu dedicar um tempo para ler esta monografia e me aconselhar na escrita deste TCC. E aos meus irmãos em fé Yasmin, Layana, Anderson e Camila, por suas orações, apoio e por estarem presentes nos momentos de fraqueza.

Por fim, agradeço a todos que contribuíram para esta caminhada. Muitos mais mereceriam ser mencionados aqui, e guardarei uma eterna gratidão pelo apoio que me deram durante esta jornada.

RESUMO

A representação visual de algoritmos tem sido um recurso popular em sala de aula para facilitar seu entendimento bem como aproximar o nível de abstração e decomposição. No entanto, há muitos casos em que simples desenhos, diagramas e vídeos não-interativos não são o suficiente para um processo significativo de aprendizagem. Isso porque eles não são capazes de demonstrar o passo a passo do algoritmo ou adaptar a entrada para mostrar casos específicos. Mesmo IDE's também permitem algum nível de depuração de código em tempo de execução, mas com muitas limitações. Outro fator importante quando falamos do uso dessas animações é que muitos desses métodos não são interativos o suficiente para prender a atenção dos alunos. Tais fatores talvez exijam métodos mais interativos e lúdicos para esse propósito. Este trabalho investiga o emprego da Realidade Aumentada para criar experiências imersivas para o aprendizado de algoritmos com vetores. O aplicativo CardSortRA faz uso de cartas físicas para oferecer uma projeção digital de uma máquina computacional que simula o funcionamento de algoritmos clássicos com vetores. Uma experiência controlada em sala de aula mostra resultados promissores e potencial para beneficiar o processo de ensino.

Palavras-chave: realidade aumentada; visualização de algoritmos; pensamento computacional; informática e educação.

ABSTRACT

The visual representation of algorithms has been a popular classroom resource to facilitate understanding as well as to bridge the gap in abstraction and decomposition levels. However, there are many cases where simple drawings, diagrams, and non-interactive videos are not sufficient for meaningful learning. This is because they are unable to demonstrate the step-by-step execution of algorithms or adapt inputs to showcase specific cases. Even IDEs allow some level of runtime code debugging, but with many limitations. Another important factor in the use of such animations is that many of these methods are not interactive enough to capture students' attention. These factors may call for more interactive and playful approaches for this purpose. This work investigates the use of Augmented Reality to create immersive experiences for learning algorithms with arrays. The CardSortAR application uses physical cards to provide a digital projection of a computational machine that simulates the operation of classic array-based algorithms. A controlled classroom experiment shows promising results and potential benefits for the teaching process.

Keywords: augmented reality; algorithm visualization; computational thinking; computer science education.

LISTA DE FIGURAS

Figura 1: Exemplo de anotação 15
Figura 2: “Sorting Out Sorting (1980)” 16
Figura 3: “Quicksort with Hungarian, folk dance” 16
Figura 4: LEPA 17
Figura 5: JavaMy 18
Figura 6: Objetos em uma cena 21
Figura 7:exemplo de ciclo de vida renderização 23
Figura 8: ciclo de vida de uma aplicação Unity 24
Figura 9: Exemplo de código errado no desenvolvimento de jogos 25
Figura 10: Exemplo de código de desenvolvimento de jogos 26
Figura 11: Modo de pensar animação 26
Figura 12: Exemplo de sequência de animações em jogos 27
Figura 13: Esboço Inicial do Tabuleiro 28
Figura 14: Circuito para execução de algoritmos 29
Figura 15: Registradores do circuito 29
Figura 16: Processador 30
Figura 17: vetor de *inputs* 30
Figura 18: Cartas do sistema 31
Figura 19: Demonstração das entradas 31
Figura 20: Visor de código 32
Figura 21: Diagrama de caso de uso 32
Figura 22: Diagrama de Atividades 34
Figura 23: criação das cartas na Vuforia 35
Figura 24: Pontos de referência de Vuforia 35
Figura 25: Diagrama de Atividades Animação 37
Figura 26: Busca Linear 41
Figura 27: Busca pelo Maior Elemento. 41
Figura 28: Gráfico de experiência dos respondentes 43
Figura 29: Gráfico de respostas sobre confiança nos conhecimentos em programação 44
Figura 30: Gráfico de respostas sobre avaliação do CardSortRA 46
Figura 31: Gráfico dos impactos em sala de aula do CardSortRA 49
Figura 32: Gráfico de Comparação da confiança antes e depois do uso do aplicativo 50

LISTA DE TABELAS

Tabela 1: Popularidade de ferramentas em aplicativos educacionais 20
Tabela 2: Auto avaliação dos conhecimentos do programador 40
Tabela 3: Afirmações usadas na quarta seção 42
Tabela 4: Sugestões e feedbacks 47

LISTA DE ABREVIATURAS E SIGLAS

FPS	Frames Per Second
IDE	Integrated Development Environment
LEPA	Learning Environment for Programs and Algorithms
MVP	Mínimo Produto Viável
RA	Realidade aumentada
SDK	Software Development Kit
UI	Interface do Usuário

SUMÁRIO

1. INTRODUÇÃO **12**

2. PRIMEIRO CAPÍTULO: REFERENCIAL TEÓRICO **14**

2.1 VISUALIZAÇÃO DE ALGORITMOS 14

2.2 REALIDADE AUMENTADA 18

2.2.1. Processamento de realidade aumentada **18**

2.2.1.1 Inputs 19

2.2.1.2 Módulo de processamento 19

2.2.1.3 Outputs 19

2.2.2 Ferramentas de desenvolvimento em realidade aumentada **19**

2.2.2.1 Unity Engine 20

2.2.2.2 Vuforia 21

2.3 AUTÔMATOS NAS ANIMAÇÕES DE JOGOS 22

2.3.1 Funcionamento da renderização de uma Engin*e* *22*

2.3.2 Autômatos **27**

3. SEGUNDO CAPÍTULO: CardSortRA **28**

3.1 MODELAGEM DO SISTEMA 32

4.TERCEIRO CAPÍTULO: AVALIAÇÃO E RESULTADOS OBTIDOS **39**

4.1 APLICAÇÃO EM SALA DE AULA 39

4.1.1 Método de avaliação **39**

4.1.2 Aplicação do formulário **39**

4.1.2.1. Primeira seção do formulário 40

4.1.2.2. Segunda seção do formulário 40

4.1.2.3 Terceira seção do formulário 42

4.1.2.4 Quarta seção do formulário 42

4.2 AVALIAÇÃO DOS RESULTADOS 43

4.2.1. Aprofundando no nível de conhecimento dos entrevistados **43**

4.2.2 Avaliação do aplicativo **45**

4.2.2.1 Engajamento 45

4.2.2.2 Reusabilidade 45

4.2.2.3 Clareza nas explicações 45

4.2.2.4 Percepção no aprendizado 46

4.2.2.5 Sugestões e feedbacks 46

4.2.3 Avaliação dos impactos do aplicativo no contexto de uma sala de aula **47**

4.2.4 Comparação na percepção de confiança **49**

5. CONSIDERAÇÕES FINAIS 51

REFERÊNCIAS
52

1. INTRODUÇÃO

O aprendizado de programação, especialmente para iniciantes, apresenta desafios significativos, sendo a interpretação de algoritmos um dos principais obstáculos. Conceitos como laços de repetição e estruturas condicionais exigem do estudante a capacidade de abstrair o fluxo de execução, o que nem sempre é intuitivo. Para mitigar essa dificuldade, uma abordagem comum é a utilização de recursos visuais, como vídeos, diagramas e animações, que auxiliam na compreensão da lógica por trás dos algoritmos ao tentar demonstrar seu funcionamento.

Apesar dessas estratégias, ainda existem limitações quanto à retenção da atenção dos alunos e à clareza nas representações dos passos de execução ou em nuances e casos específicos dos algoritmos. Nesse contexto, ferramentas capazes de demonstrar de forma interativa e visual o funcionamento dos algoritmos por meio da computação, tornam-se alternativas promissoras, além de facilitar a explicação, elas evidenciam processos que muitas vezes ficam ocultos pela própria linguagem de programação, impactando diretamente no aprendizado.

A realidade aumentada (RA), por sua vez, apresenta grande potencial educacional, pois, quando integrada a projetos pedagógicos, proporciona experiências imersivas e interativas, capazes de aumentar o engajamento e tornar o aprendizado mais acessível e atrativo para os estudantes. Esse aspecto somado com o processo de animações dos algoritmos pode ser de grande ajuda no processo de ensino e aprendizagem.

Diante disso, este trabalho tem como objetivo demonstrar uma ferramenta que auxilie no ensino de algoritmos por meio de sua demonstração visual e compartilhar um pouco das experiências obtidas durante o processo de desenvolvimento e teste em sala de aula. Para atingir esses objetivos, foi desenvolvido um aplicativo educacional chamado CardSortRA utilizando a plataforma Unity e a biblioteca Vuforia, permitindo a execução interativa de algoritmos a partir de entradas definidas pelos próprios usuários.

Com base na ferramenta proposta, avaliamos seu potencial educacional por meio de um teste em sala de aula com uma turma do primeiro período do curso de Ciência da Computação. Utilizando como parâmetro de avaliação a Escala de Likert para medir os seguintes elementos subjetivos: engajamento, motivação e confiança, o estudo verifica se o aplicativo impactou positivamente no aprendizado, conforme a percepção dos estudantes. Este

estudo serve como indicativo para avaliar o potencial educacional da realidade aumentada para visualização de algoritmos.

O presente trabalho está organizado da seguinte forma. O primeiro capítulo apresenta o referencial teórico, abordando conceitos de visualização de algoritmos, realidade aumentada e animação de autômatos. O segundo capítulo descreve a ferramenta proposta. O terceiro capítulo detalha o método de avaliação do sistema e os resultados obtidos, além da apresentação das considerações finais.

2. REFERENCIAL TEÓRICO

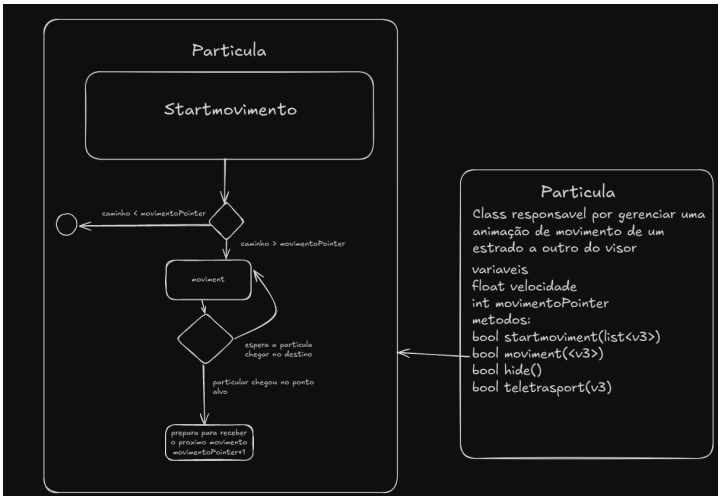
O referencial teórico deste trabalho concentra-se em três temas fundamentais para sua execução: visualização de algoritmos, realidade aumentada e autômatos. A visualização de algoritmos é um assunto popular no ensino de programação, já que permite entender de forma simplificada, processos que são muitas vezes difíceis de abstrair no ensino tradicional de algoritmos, sem contar que ela permite mostrar de forma lúdica etapas que costumam ficar ocultas, principalmente por linguagens mais atuais. Somando-se a esses benefícios, a realidade aumentada pode atuar nesse processo como uma ferramenta para ampliar a imersão do estudante nas ferramentas tradicionais de educação. Para a elaboração deste trabalho, foi necessária a compreensão de autômatos, uma vez que o núcleo de desenvolvimento consiste em uma máquina de estados capaz de computar os algoritmos.

2.1 Visualização de algoritmos

A visualização de algoritmos é uma forma de comunicar e compreender métodos computacionais, consistindo na representação desses processos de forma visual por meio de diagramas, imagens e ilustrações que facilitam a compreensão do algoritmo. Baecker (1998) menciona que,

> Como programadores profissionais, professores e estudantes de ciência da computação frequentemente usam imagens como auxílio para conceber, expressar e comunicar algoritmos. Instrutores costumavam cobrir os quadros-negros e a si com giz ao desenhar diagramas intricados de estruturas de dados e fluxo de controle (Baecker, 1998, p. 369).

Um exemplo dessa abordagem é ilustrado na anotação(Figura 1).



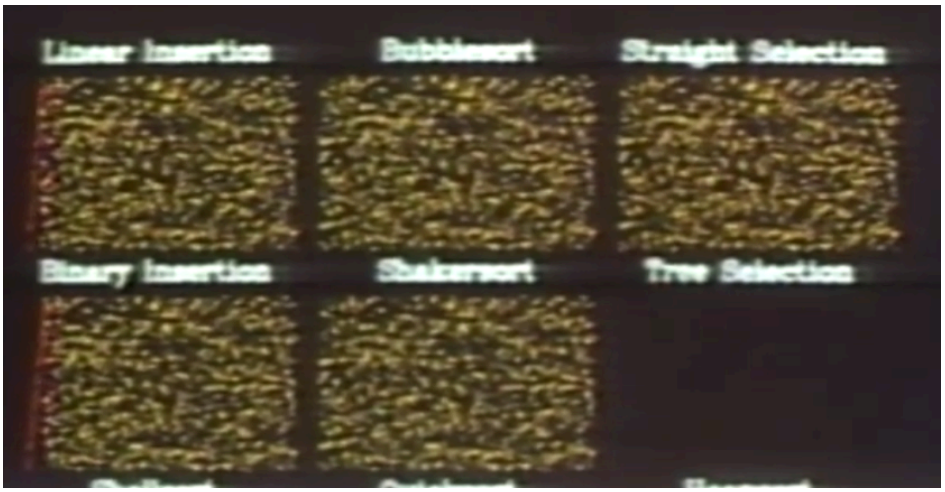
Fonte: Elaborado pelo autor.

Porém, frequentemente essa comunicação é falha devido a erros na criação ou à incapacidade de estruturar fluxos algorítmicos corretos. Ademais, como o educador Baecker (1998) descreveu em seu artigo:

> Infelizmente, o comportamento de um programa não pode ser descrito por um desenho estático; ele requer uma sequência dinâmica. Precisamos rastrear o fluxo de controle, vincular variáveis, ligar ponteiros e alocar memória. Precisamos executar processos que imitam os da máquina. É difícil para nós ensinarmos essas sequências dinâmicas diretamente. Nossos desenhos são imprecisos (Baecker, 1998, p. 369).

Diante dessa insatisfação com os métodos tradicionais de representação algorítmica surgiram as primeiras tentativas de expressar essas sequências lógicas por meio animações como ‘Sorting Out Sorting’ (Figura 2), um filme proposto pelo próprio Baecker (1980) ou o famoso vídeo da apresentação de Quicksort with Hungarian, folk dance (Figura 3).

Figura 2: “Sorting Out Sorting (1980)”



Fonte: Baecker, 1998.

Figura 3: “Quicksort with Hungarian, folk dance”

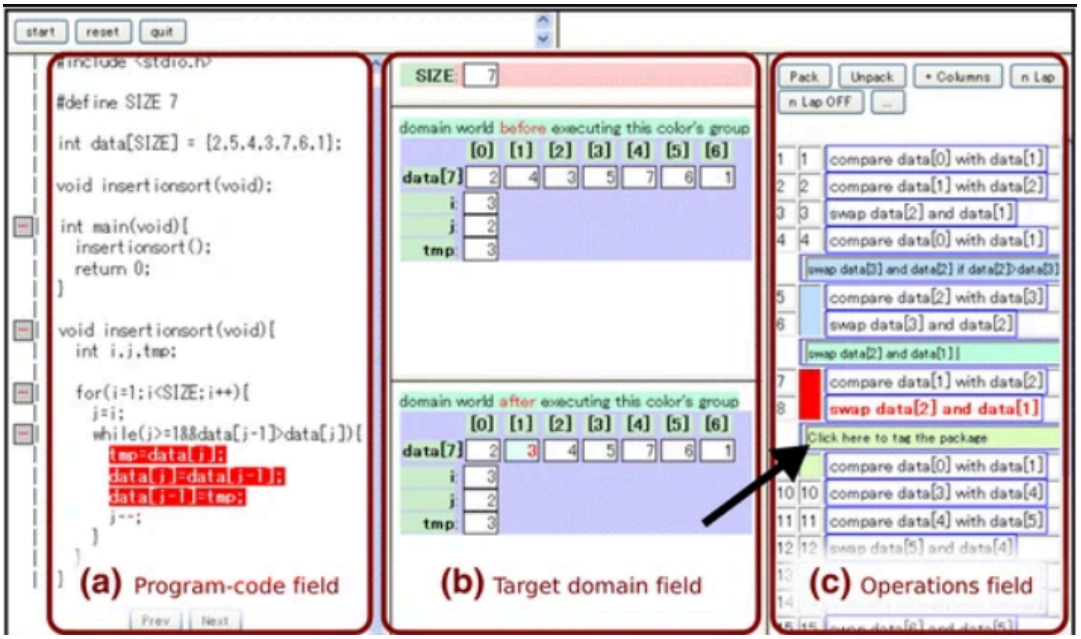
Fonte: https://www.youtube.com/watch?v=3San3uKKHgg

Apesar das propostas anteriores, Koichi Yamashita et. all. (2016) argumentava que frequentemente a simples animação por meio de vídeo não permite a demonstração integralmente do algoritmo e nem oferece a adaptabilidade necessária para o ambiente educacional, conforme destacado pelo autor:

> Normalmente, as estruturas de dados visualizadas são exibidas para os alunos, mostrando-lhes slides e/ou filmes produzidos com software de apresentação e edição de vídeo. No entanto, esses materiais não podem ser usados para certas atividades de aprendizagem, como quando os alunos são obrigados a observar o comportamento do programa, onde os dados de entrada são alterados individualmente, porque os dados de entrada são fixos (Yamashita et all., 2016, p. 03).

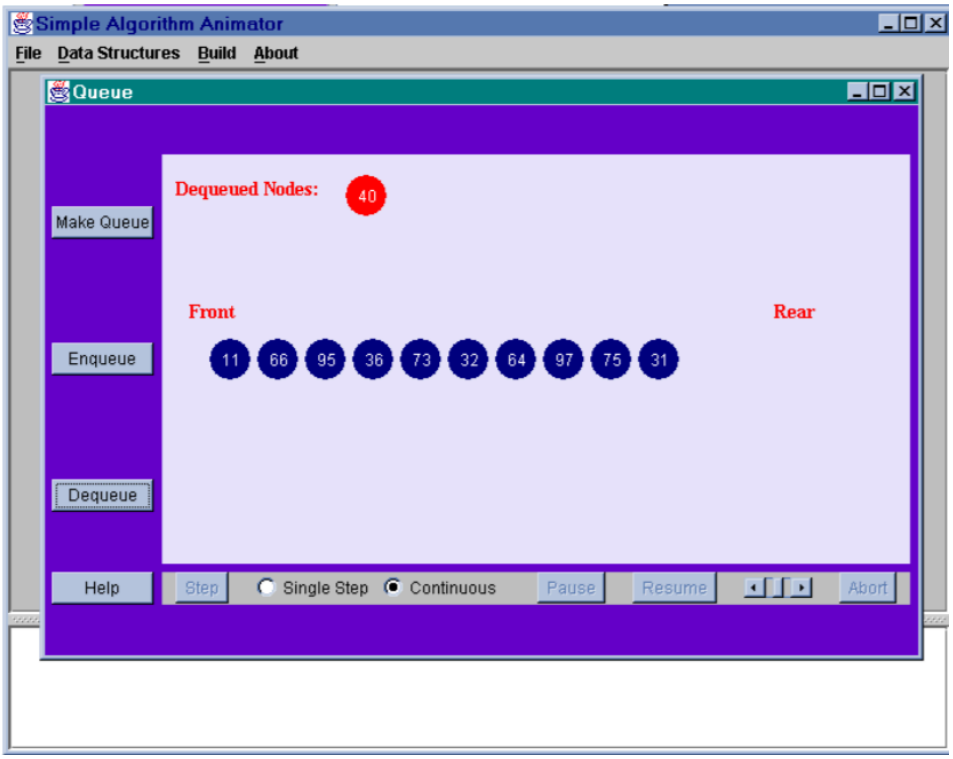
Como solução para esse problema, surgiram sistemas que permitem visualizar algoritmos em tempo de execução por meio de animações que editáveis durante a execução, como a plataforma proposta pelo próprio Yamashita (2016) LEPA - Learning Environment for Programs and Algorithms, conforme ilustrado na Figura 4. Outro exemplo pode ser visto na Figura 5, que exibe o JavaMy (SOBH, 2001). A plataforma LEPA concentra-se principalmente na demonstração da memória em tempo de execução enquanto o JavaMy é mais lúdica, adota uma abordagem mais lúdica, animando os passos dos algoritmos usando elementos visuais claros – como círculos com os valores das entradas como elementos da estrutura de dados, além de mostrar os movimentos desses objetos que correspondem à movimentação de memória durante a execução do algoritmo.

Figura 4: LEPA



Fonte: Practices of algorithm education based on discovery learning using a program visualization system

Figura 5: JavaMy



Fonte: A Tool for Data Structure Visualization and User-Defined Algorithm Animation

2.2 REALIDADE AUMENTADA

A Realidade Aumentada (RA), também conhecida pela sigla AR (do inglês Augmented Reality), conforme Azuma (2001), pode ser definida como “um sistema que suplementa o mundo real com objetos virtuais gerados por computador, o qual combina objetos reais e virtuais no ambiente real; executa interativamente em tempo real; alinha objetos reais e virtuais entre si”. Complementando essa definição, a Unity Technologies [s.d.] afirma: “A realidade aumentada é a sobreposição de conteúdo criado digitalmente sobre o mundo real. A realidade aumentada — ou "AR" — permite que o usuário interaja tanto com o mundo real quanto com elementos ou ampliações digitais”.

2.2.1. Processamento de realidade aumentada

O processo de suplementação é normalmente feito por um sistema que possui 3 componentes principais: *Inputs*, Módulo de Processamento e *Outputs*.

2.2.1.1 *Inputs*

Os *inputs* (ou entradas de dados) consistem em mecanismos capazes de captar a realidade e transformá-la em estímulos ou dados. O input mais comum em realidade aumentada é o visual, obtido por meio de câmeras. No entanto, a captação de dados também pode ocorrer por outros meios sensoriais, como microfones (*input* auditivo) e acelerômetros (*input* cinestésico)

2.2.1.2 Módulo de processamento

Um módulo de processamento responsável por tratar os dados de entrada, com as seguintes funções:

1. Processar a posição dos objetos em relação ao ambiente real;
2. Identificar as interações com o usuário;
3. Refletir essas ações nos objetos digitais, conferindo sentido ao contexto da aplicação.

2.2.1.3 *Outputs*

Responsáveis por projetar os objetos no ambiente real, os *outputs* (ou saídas) podem ser exemplificados por áudio tridimensional e objetos renderizados por câmeras.

É importante ressaltar que um dos principais desafios nos projetos de realidade aumentada atualmente encontra-se nas etapas de *inputs* (entradas) e *outputs* (saídas), pois estes estão diretamente vinculados ao hardware do dispositivo utilizado, o que pode comprometer a qualidade da experiência. Por exemplo, as câmeras de smartphones não são projetadas para medir distâncias com precisão, o que dificulta o posicionamento preciso de objetos em superfícies complexas.

2.2.2 Ferramentas de desenvolvimento em realidade aumentada.

No contexto do desenvolvimento de aplicativos educacionais utilizando realidade aumentada, as ferramentas mais empregadas, conforme apontado pela pesquisa de Herpich et al. (2019, p. 1601), podem ser vistas na Tabela 1.

Tabela 1: Popularidade de ferramentas em aplicativos educacionais
Fonte: Herpich et al. (2019, p. 1601)

Ferramenta / Linguagem de programação	Quantidade de artigos.
Unity	5
XCode	5
AR content platforms: Layar	5
Junaio	4
ARToolKitPlus	4
Vuforia	3

Os próximos tópicos são focados nas ferramentas Unity Engine e Vuforia, já que foram as utilizadas na criação desse projeto.

2.2.2.1 Unity Engine

Conforme o pesquisador Haas (2025),

> Unity (comumente conhecida como Unity3D) é uma *game engine* e um ambiente de desenvolvimento integrado (IDE) para a criação de mídias interativas, tipicamente *vídeogames*. Como afirmou o CEO David Helgason, Unity 'é um conjunto de ferramentas usado para construir jogos, e é a tecnologia que executa os gráficos, o áudio, a física e as interações na rede'. Unity é famosa por suas capacidades de prototipagem rápida e pelo grande número de plataformas para as quais permite publicação (Haas, 2025, p. 1).

O principal destaque da Unity no mercado de jogos digitais reside em sua ampla adoção. Segundo Holfeld (2025, p. 3), a engine detém aproximadamente 60% do mercado, enquanto dados da Astute Analytica India Pvt. Ltda. (DATE) indicam uma participação de 51%. A Unity se sobressai principalmente por sua flexibilidade e pela diversidade de ferramentas integradas ou acessíveis por meio de *plugins*, oferecendo suporte nativo ao desenvolvimento em 2D, 3D e em realidade aumentada (RA), com integração ao ARCore. Além disso, a plataforma conta com uma vasta quantidade de tutoriais disponíveis e uma comunidade de desenvolvedores ativa, o que facilita a aprendizagem e acelera o desenvolvimento de projetos.

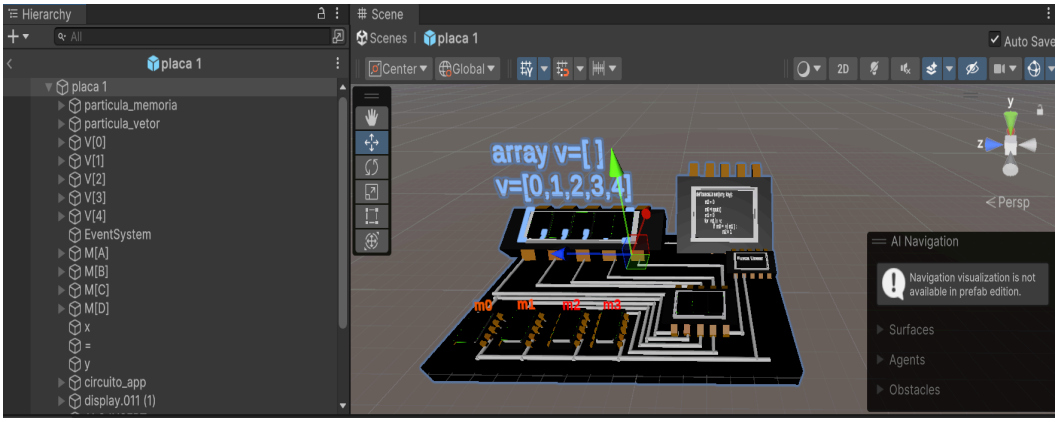
Conforme descrito pela documentação oficial da Unity (Unity Technologies, [s.d.]) o funcionamento da Unity se baseia em dois objetos principais:

GameObjects: Os *GameObjects* são os objetos fundamentais nas cenas da Unity e podem representar personagens, adereços (*props*), cenários, câmeras, pontos de passagem (*waypoints*), entre outros. A funcionalidade de um *GameObject* é definida pelos componentes que lhe são atribuídos. Todos os objetos dentro de uma cena são *GameObjects*. Em contraste, os do projeto são arquivos-fonte adicionados às cenas, com scripts em C#, texturas, materiais, arquivos de modelos 3D e *prefabs*. Os *GameObjects* existem em ambientes 2D ou 3D, chamados de cenas.

As Cenas: As cenas são os objetos onde se trabalha com o conteúdo na Unity. Elas são *assets* (recursos) que contêm toda ou parte de um jogo/aplicação. Por exemplo, é possível construir um jogo simples em uma única cena, enquanto, em um jogo mais complexo, pode-se utilizar uma cena por nível – cada uma com seus próprios ambientes, personagens, obstáculos, decorações e interface do usuário (UI).

Na Figura 6 temos o exemplo de uma cena com uma hierarquia de *Gameobjects* na qual o objeto *placa* é composto por diversos outros *Gameobjects*, formando assim um objeto complexo.

Figura 6: Objetos em uma cena



Fonte: Elaborado pelo autor.

2.2.2.2 Vuforia

De acordo com a própria desenvolvedora,

> O Vuforia Engine é a plataforma mais amplamente utilizada para o desenvolvimento de Realidade Aumentada (RA), com suporte para a maioria

> dos *smartphones*, *tablets* e dispositivos vestíveis. Os desenvolvedores podem facilmente adicionar funcionalidades avançadas de visão computacional a aplicativos Android, iOS e UWP, criando experiências de RA que interagem de forma realista com objetos e com o ambiente.

A Vuforia possui duas aplicações principais: o Vuforia SDK e o Vuforia Desktop Tools.

O Vuforia SDK (*Software Development Kit*), desenvolvido para as linguagens C e C#, é compatível com as plataformas Windows e Mobile. Pode ser integrado à Unity, facilitando processos como reconhecimento de alvos (imagens), posicionamento de objetos em planos e monitoramento por meio de ferramentas que simplificam o processamento em realidade aumentada.

Já o Vuforia Desktop Tools é um software específico para a criação de aplicações simples em realidade aumentada. Sua principal característica é a capacidade de gerenciar (criar) *targets*, realizar reconhecimento de ambiente e posicionar objetos em realidade aumentada.

2.3 AUTÔMATOS NAS ANIMAÇÕES DE JOGOS

Ao programar aplicações como páginas *web*, jogos e aplicativos *mobile*, é comum depararmos- nos com ciclos de vida e de renderização específicos das ferramentas utilizadas no desenvolvimento, o que afeta significativamente a forma como as programamos. Isso ocorre porque, nessas ferramentas não temos o controle direto sobre o ciclo de vida da aplicação, mas sim sobre bibliotecas e recursos que interagem com ele – aspecto vantajoso, pois abstrai do programador diversos processos que seriam complexos de implementar manualmente.

Contudo, essa característica influencia a abordagem de desenvolvimento do código, afastando-o da programação sequencial e linear e aproximando-o de um paradigma orientado a estados e eventos. Esse comportamento, por sua vez, assemelha-se ao funcionamento de autômatos.

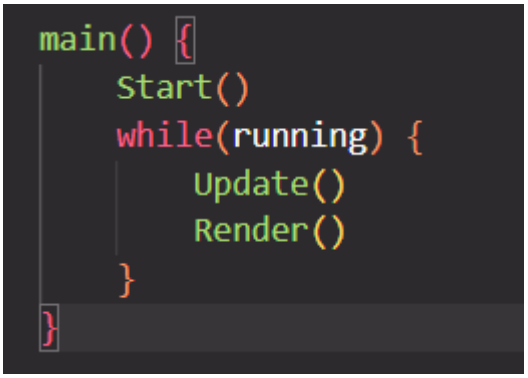
2.3.1 Funcionamento da renderização de uma *engine*

De modo geral, o ciclo de vida da maioria das aplicações gráficas — incluindo jogos, aplicações interativas e páginas web — pode ser dividido em três etapas principais:

1. **Start( ):** Responsável pelo carregamento e inicialização dos dados quando o objeto ou cena é iniciado.
2. **Update( ):** Executado continuamente para atualizar dados e processar a lógica durante a execução.
3. **Render( ):** Responsável por renderizar os elementos visuais na tela.

Em pseudocódigo, esse ciclo pode ser representado conforme mostra a Figura 7.

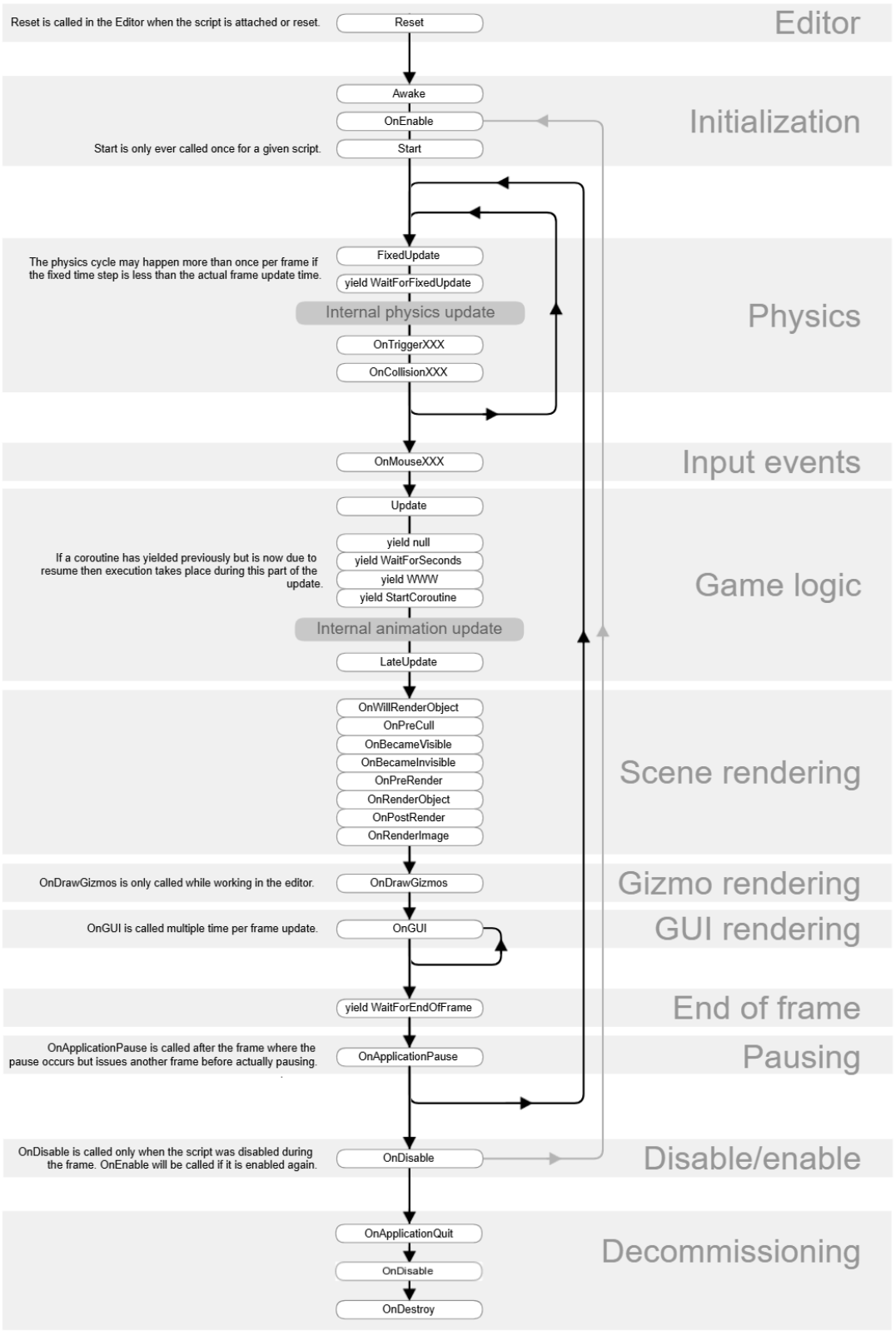
Figura 7:exemplo de ciclo de vida renderização



Fonte: Elaborado pelo autor.

Um exemplo desse funcionamento está ilustrado na Figura 8 abaixo, que demonstra o ciclo de vida de um objeto na Unity Engine.

Figura 8: ciclo de vida de uma aplicação Unity



Fonte: https://docs.unity3d.com/

A maneira como esse ciclo afeta a programação inviabiliza o uso de estruturas de repetição no código para ações que deveriam ser graduais. Para exemplificar de forma mais prática, pode-se demonstrar esse caso com um laço *while*, que executa continuamente até que sua condição se torne falsa. No código da Figura 9, esse laço é utilizado para mover o objeto ao longo do eixo x de um extremo a outro da tela.

Figura 9: Exemplo de código errado no desenvolvimento de jogos

Fonte: Elaborado pelo autor.

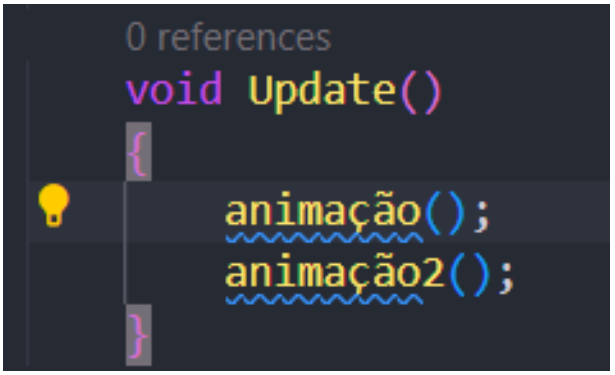
Contudo, o resultado real não seria uma animação, mas sim o programa ficando retido na função Update até que o laço *while* fosse finalizado, com o objeto já em sua posição final. Essa situação poderia causar erros durante a execução, como tela preta ou baixa taxa de FPS (*frames per second*), que corresponde à medida de quantas vezes o aplicativo executa o loop principal. A implementação adequada seria semelhante à apresentada na Figura 10.

Figura 10: Exemplo de código de desenvolvimento de jogos

Fonte: Elaborado pelo autor.

Apesar de parecer uma pequena mudança, isso implica que, como não há controle sobre o fluxo direto da aplicação, é necessário abordar o processamento em jogos com uma perspectiva diferente daquela usualmente ensinada em ambientes acadêmicos. Outro exemplo pode ser observado na Figura 11, que ilustra como tradicionalmente se conceberia a execução sequencial de animações.

Figura 11: Modo de pensar animação



Fonte: Elaborado pelo autor.

Contudo, a forma correta de utilizar animações seria a demonstrada na Figura 12, que se aproxima progressivamente do modo como operam as máquinas de estados: ao finalizar uma animação, altera-se o estado, que fica preparado para a próxima animação. Portanto, ao tratarmos de animações complexas em jogos - especialmente as condicionadas a determinadas circunstâncias - o contexto dos autômatos.

Figura 12: Exemplo de sequência de animações em jogos

```
void update(){
    if(estado == "animação1"){
        if(animação()){
            estado = "animação2";
        }
    }else if(estado == "animação2"){
        if(animação2()){
            estado = "fim";
        }
    }else{
        //...
    }
}
```

Fonte: Elaborado pelo autor.

2.3.2 Autômatos

Segundo material apresentado por Newton José Vieira (s.d.),

> Um autômato finito determinístico é uma estrutura matemática constituída por três tipos de entidades: um conjunto de estados, um alfabeto e um conjunto de transições. Dentre os estados, destaca-se um como estado inicial, bem como um subconjunto de estados chamados de estados finais.

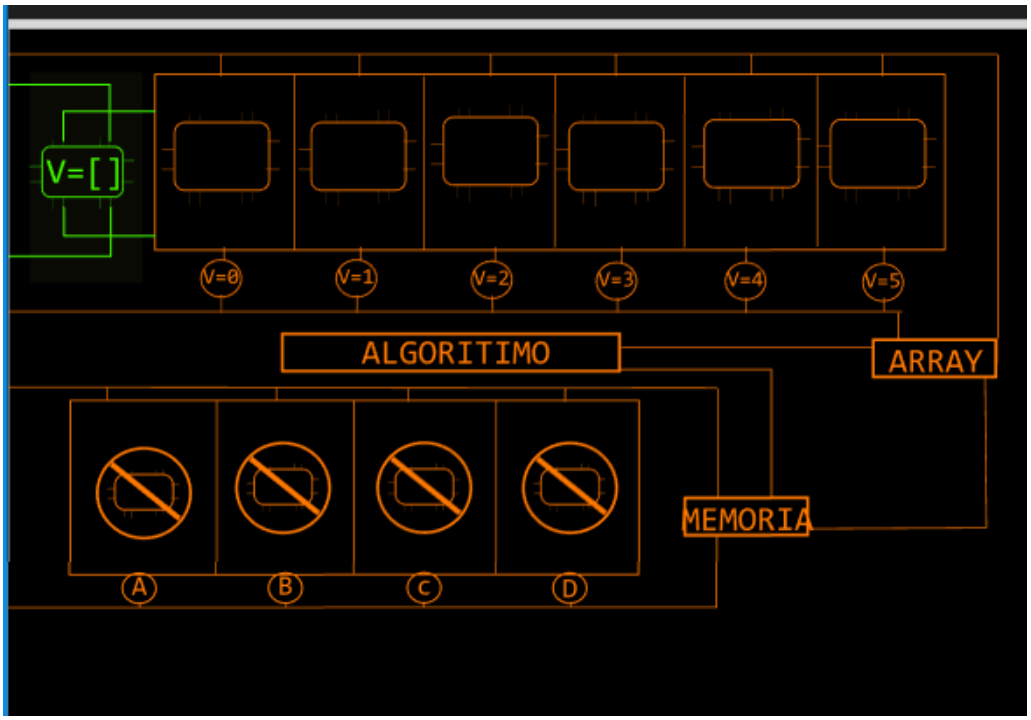
Essa definição aplica-se ao contexto de animações de algoritmos, em que: o conjunto de estados é representado pelas possíveis animações e operações executadas pelo autômato. O alfabeto corresponde a lista de instruções do algoritmo e as funções de transição representam a lógica do algoritmo propriamente dita, na qual um conjunto de instruções é processado como uma fila de ações definidas e finitas. Desse modo, o início é representado pelo estado inicial do algoritmo e o estado final ocorre quando a fila de algoritmos está vazia, o que acrescenta um elemento de valores lógicos aos objetos das animações. Esse processo assemelha-se consideravelmente ao que ocorre em compiladores, permitindo que a criação das animações represente também uma visualização mais clara do que acontece em nível próximo ao processador.

3 CardSortRA

A proposta deste trabalho é desenvolver uma ferramenta que auxilie no estudo de algoritmos por meio da representação gráfica e lúdica de sua execução em um ambiente imersivo de realidade aumentada, com o propósito de aproximar o nível de abstração e facilitar a compreensão sobre o funcionamento de algoritmos clássicos em vetores.

Como esboço inicial, foi pensando no tabuleiro da Figura 13, no qual os algoritmos poderiam ser executados sobre ele como um jogo de cartas. Tal esboço foi pensado de forma que houvesse dois elementos principais: um contendo as cartas do desafio, chamado de vetor; e outro chamado de memória. Pela troca das cartas de lugar entre o vetor e a memória, seriam executados os algoritmos.

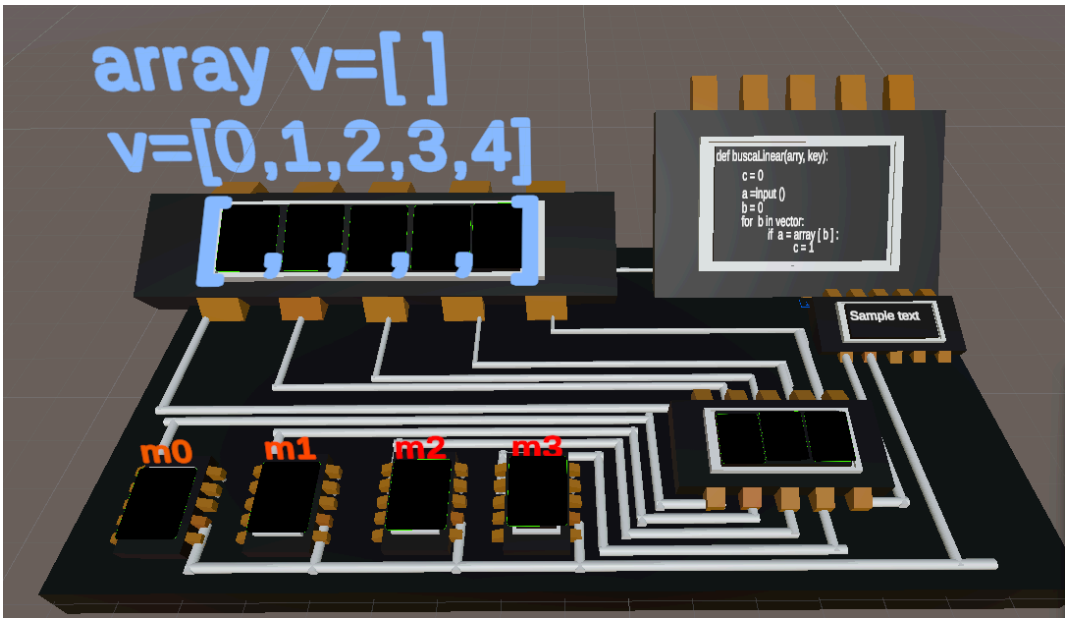
Figura 13: Esboço Inicial do Tabuleiro



Fonte: Elaborado pelo autor.

Com o avanço dessa ideia, foi feita uma melhoria apresentada na Figura 14, que descreve a máquina abstrata utilizada para ilustrar a execução dos algoritmos. Tal máquina consiste em uma placa de circuito que possui três componentes principais criados com o intuito de se assemelhar ao que realmente ocorre na execução de um algoritmo. Tais partes são explicadas em maiores detalhes a seguir.

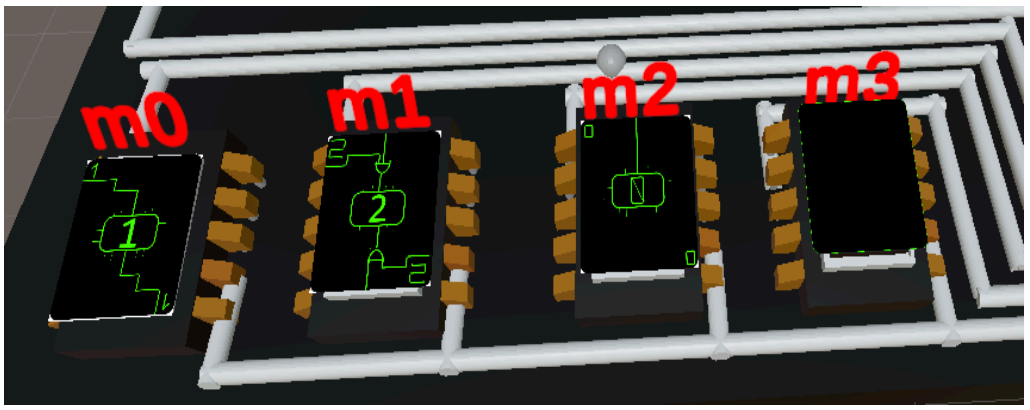
Figura 14: Circuito para execução de algoritmos



Fonte: Elaborado pelo autor.

Como pode ser observado na Figura 15, os registradores são representados pelos objetos m0, m1, m2 e m3, que funcionam como variáveis ou memória auxiliar para as operações dos algoritmos. O uso dos registradores será abordado com maior profundidade no próximo capítulo.

Figura 15: Registradores do circuito

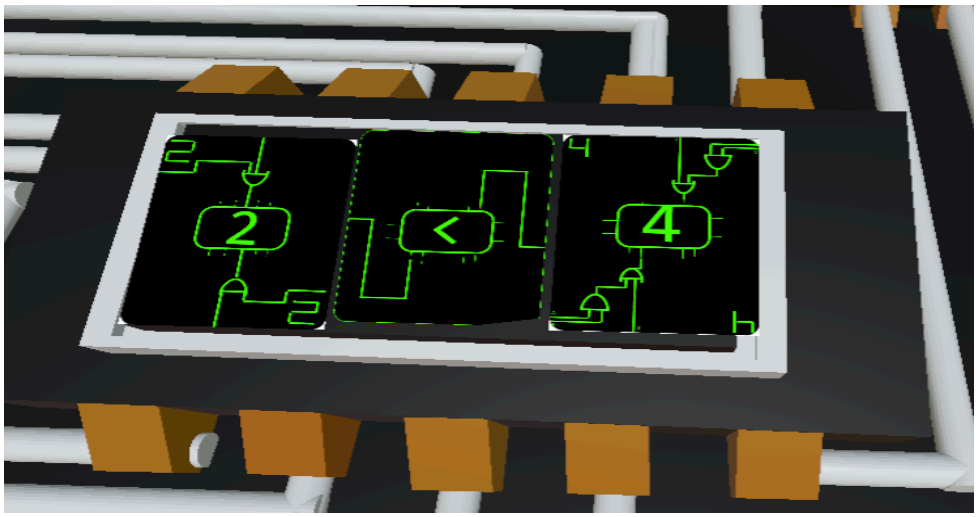


Fonte: Elaborado pelo autor.

A parte do processador corresponde à representação do núcleo de processamento de uma máquina, no qual são realizadas todas as operações lógicas. Esse núcleo é composto por áreas de texto, que representam dois valores numéricos e uma operação lógica ou matemática,

conforme ilustrado na Figura 16. Nessa figura, é apresentado o teste 'se 2 é menor que 4'. As possíveis computações serão analisadas em detalhes na próxima seção.

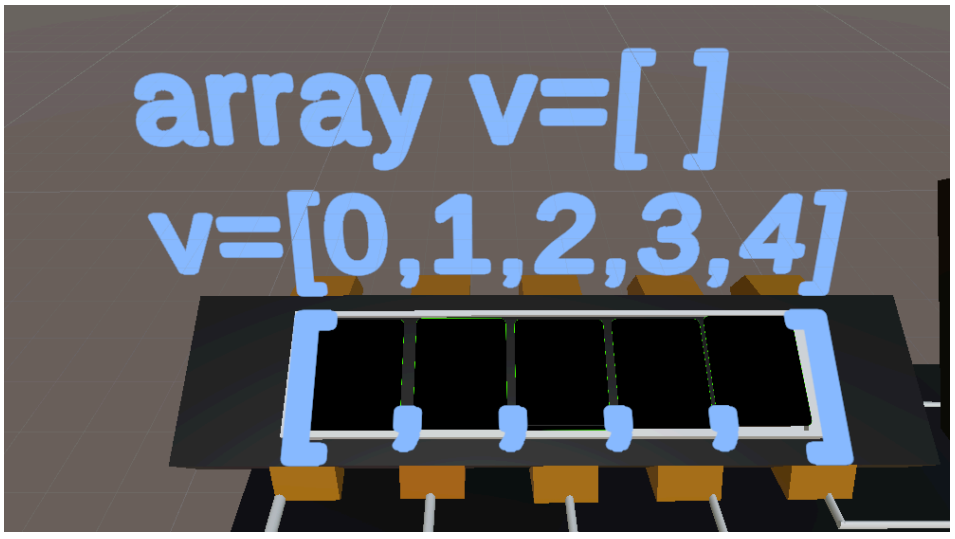
Figura 16: Processador



Fonte: Elaborado pelo autor.

Por fim, a Figura 17 exibe o vetor de entrada, utilizado para representar a instância na qual o algoritmo é aplicado. Na prática, a maioria dos problemas abordados envolve operações com vetores. Sua representação é feita por meio de um conjunto de cinco cartas indexadas de 0 a 4.

Figura 17: vetor de *inputs*

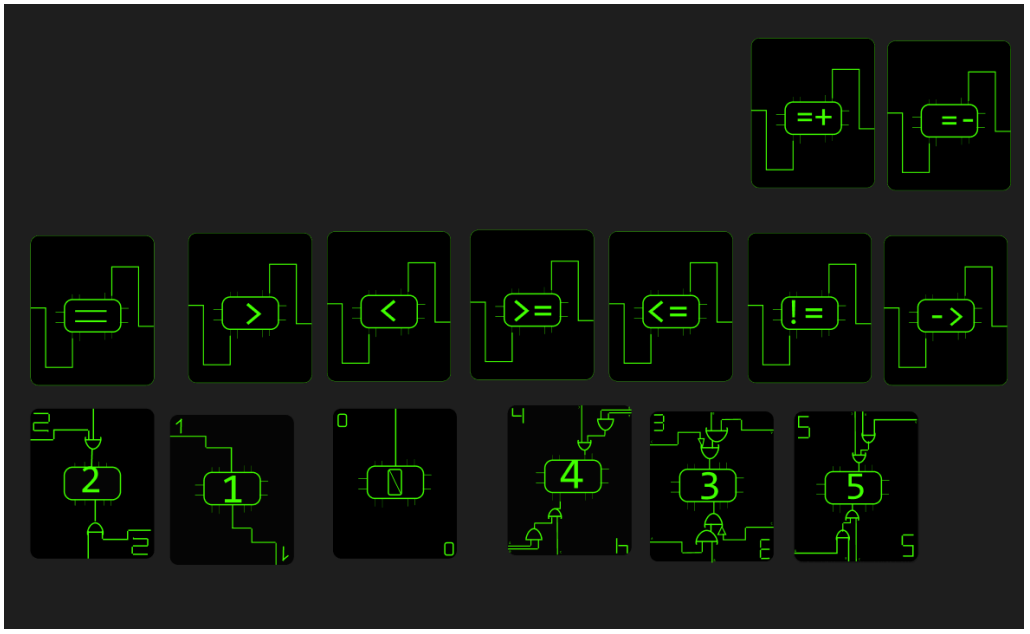


Fonte: Elaborado pelo autor.

Para tornar a experiência mais interativa e lúdica, optou-se por uma abordagem inovadora na captura das entradas do sistema, criando-se um conjunto de cartas a serem escaneadas como input do sistema.

A Figura 18 apresenta o conjunto de cartas desenvolvidas para essa finalidade. Para que uma entrada seja considerada válida, é necessário que contenha uma carta de referência e até cinco cartas numéricas, que funcionarão como dados de entrada.

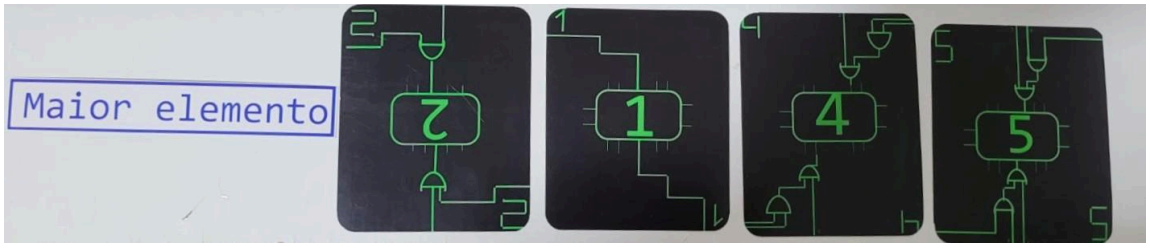
Figura 18: Cartas do sistema



Fonte: Elaborado pelo autor.

A Figura 19 exibe um exemplo de entrada válida, em que a primeira carta à esquerda funciona como referência, enquanto as demais representam os valores inteiros a serem armazenados inicialmente no vetor de entrada (neste caso: [2,1,4,5]).

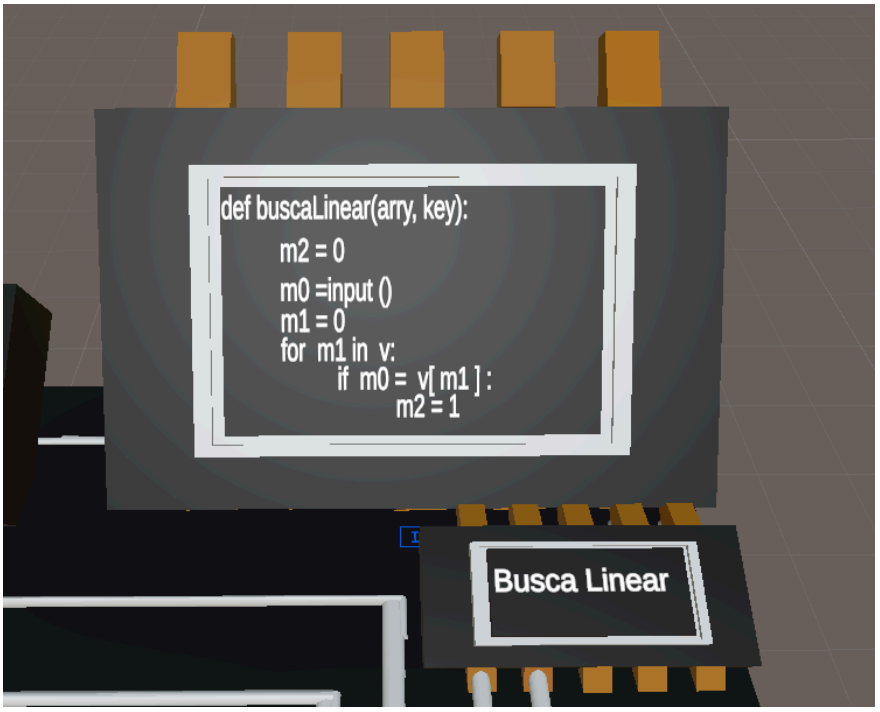
Figura 19: Demonstração das entradas



Fonte: Elaborado pelo autor.

Adicionou-se também um pequeno visor que permite ao aluno visualizar o código em execução, conforme ilustrado na Figura 20. Dessa forma, o aluno pode acompanhar o processo durante as animações e identificar qual algoritmo está sendo executado

Figura 20: Visor de código

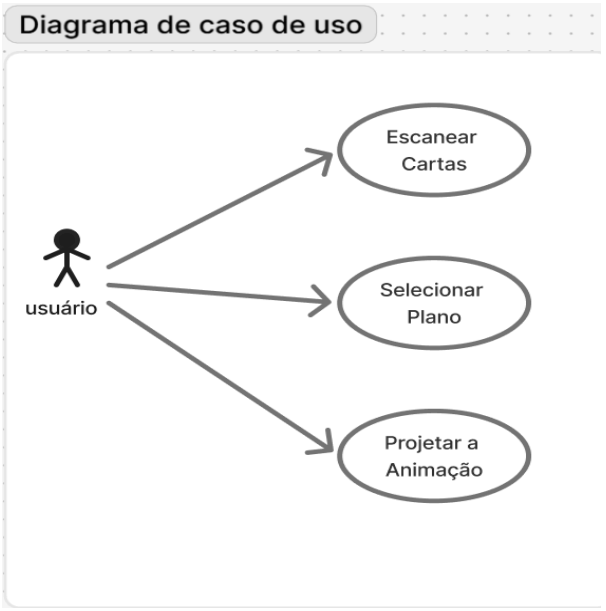


Fonte: Elaborado pelo autor.

3.1 Modelagem do Sistema

A Figura 21 ilustra um caso de uso do sistema. Em sua operação padrão, o usuário percorre três etapas principais: escaneamento das cartas, seleção dos planos e projeção da animação.

Figura 21: Diagrama de caso de uso

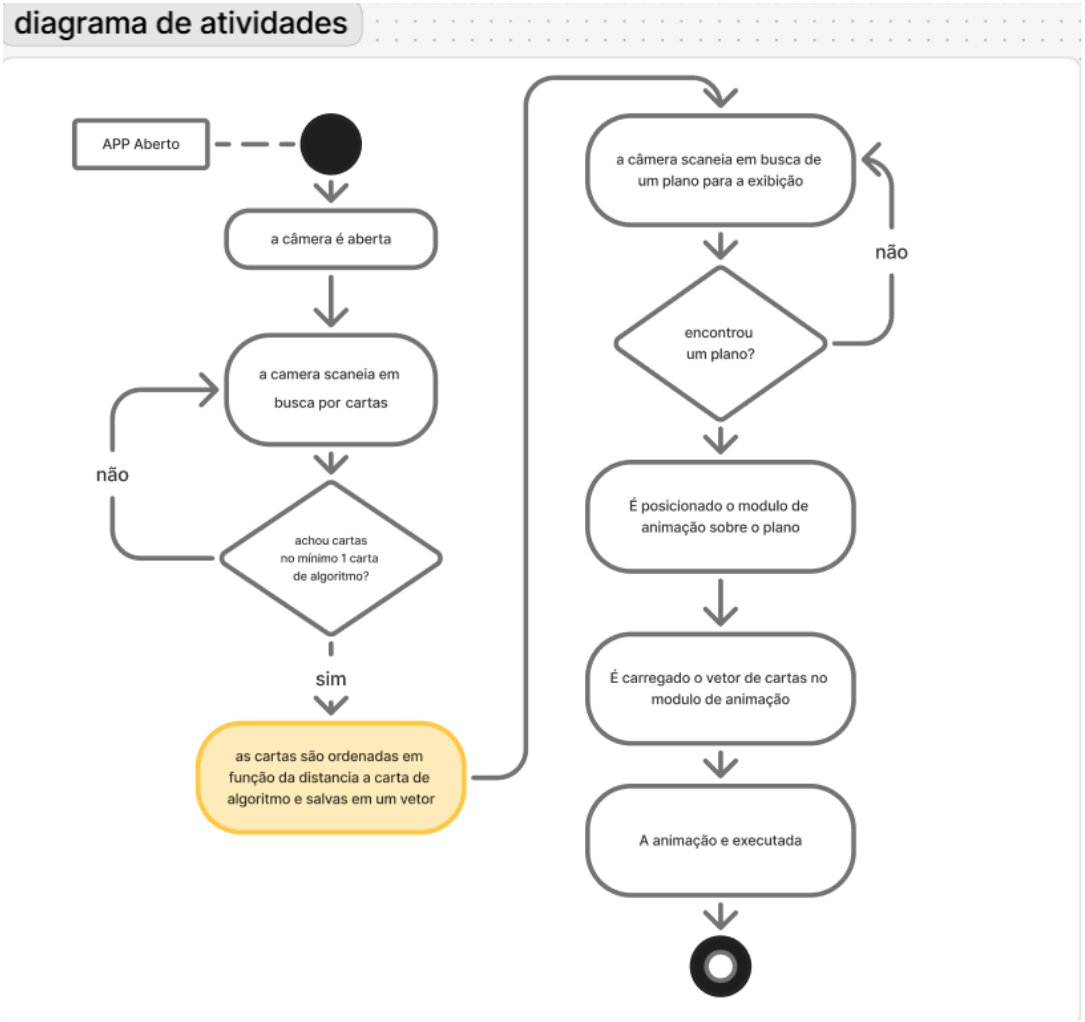


Fonte: Elaborado pelo autor.

1. **Primeira etapa - Escaneamento:** O usuário deve escanear uma sequência de cartas, composta por uma carta de algoritmo e pelo menos uma carta numérica, que serão utilizadas como entradas do vetor. Os valores das cartas são armazenados em um vetor ordenado pela distância de cada carta em relação à carta de algoritmo. A Figura 19 ilustra essa fase, mostrando um vetor inicial com os valores [2, 1, 4, 5].
2. **Segunda etapa - Posicionamento:** O usuário deve posicionar a câmera do celular para identificar uma superfície adequada para a animação. Uma vez identificado o plano, posiciona-se sobre ele o Circuito (Figura 14).
3. **Terceira etapa - Animação:** O vetor é carregado no Circuito (Figura 14) e o algoritmo é recuperado da memória (Figura 15) como uma sequência de passos de execução, para então ser processado por meio de uma máquina de estados.

Entrando em mais detalhes, a Figura 22 mostra o fluxo completo descrito anteriormente, representado em um diagrama de atividades. Em outras palavras, a Figura 21 é detalhada em cada uma das atividades subsequentes.

Figura 22: Diagrama de Atividades

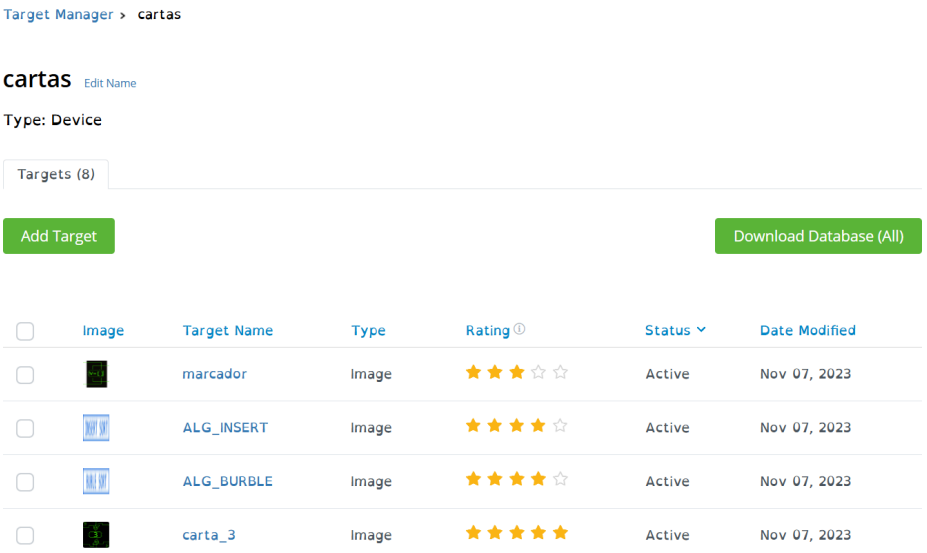


Fonte: Elaborado pelo autor.

Na primeira etapa de reconhecimento das cartas, utiliza-se a biblioteca Vuforia para Unity. Essa biblioteca possibilita a criação e identificação de objetos com base em bancos de dados desenvolvidos na plataforma Vuforia. Tais objetos podem ser interpretados pelos componentes da Unity para reconhecimento em realidade aumentada. Para criar um objeto, é necessário acessar o site da Vuforia, criar uma conta de usuário, gerar um novo banco de dados, adicionar os objetos ao banco de dados criado e exportar o banco de dados para a Unity.

A Figura 23 ilustra o processo de criação de um banco de dados na Vuforia para armazenamento das cartas. Como mostra a figura, é essencial acessar o site da empresa para desenvolver esses bancos de dados.

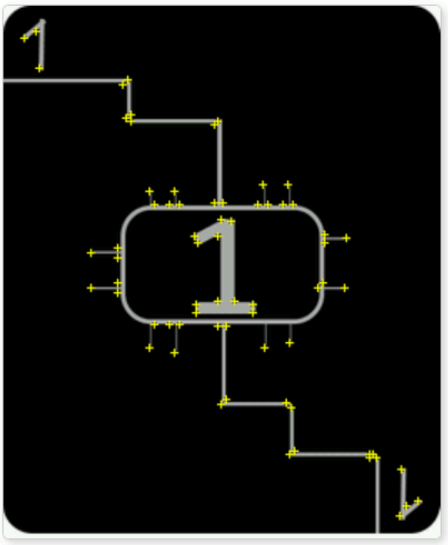
Figura 23: criação das cartas na Vuforia



Fonte: Elaborado pelo autor.

Após o envio das imagens, a plataforma Vuforia realiza um processamento que determina pontos de referência para o reconhecimento das cartas durante a execução do CardSortRA. A Figura 24 apresenta exemplos desses pontos de referência característicos das cartas. Concluído o processamento, o próximo passo consiste em baixar o banco de dados da Vuforia e configurá-lo na Unity.

Figura 24: Pontos de referência de Vuforia



Fonte: Elaborado pelo autor.

Sobre o processo de detecção das cartas, é necessário salientar um ponto nas bibliotecas. Tanto a Vuforia quanto a ARcore da própria Unity têm dificuldades em lidar com medição de distância entre o objeto projetado ou escaneado. Por isso, foi adotada como solução para esse problema a criação de um objeto de ponto de referência para poder-se ter um ponto de referência do início do vetor de entrada para pegar a ordem das cartas dispostas.

Para isso, foi usado um cálculo de distância relativa até o ponto de referência, a partir daí é feita uma ordenação das cartas detectadas baseada na distância das cartas até o ponto de referência, talvez não sendo a melhor abordagem, mas sendo uma solução viável inicialmente. Após isso, é exportado este vetor ordenado para a próxima etapa.

Na etapa de busca por um plano, é usada a biblioteca ARcore usando o objeto mid Air Stage para fazer a busca por um plano para posicionar as animações. Essa etapa é talvez a mais simples do ponto de vista de desenvolvimento porque a própria biblioteca realiza esse processo. Assim sendo, quando um plano é encontrado, o mesmo serve de entrada para a próxima etapa.

Já na etapa de animação, como citado anteriormente, há 3 objetos principais e mais um objeto auxiliar, sendo esses:

1. Memória: 04 registradores responsáveis por armazenar as informações do programa.

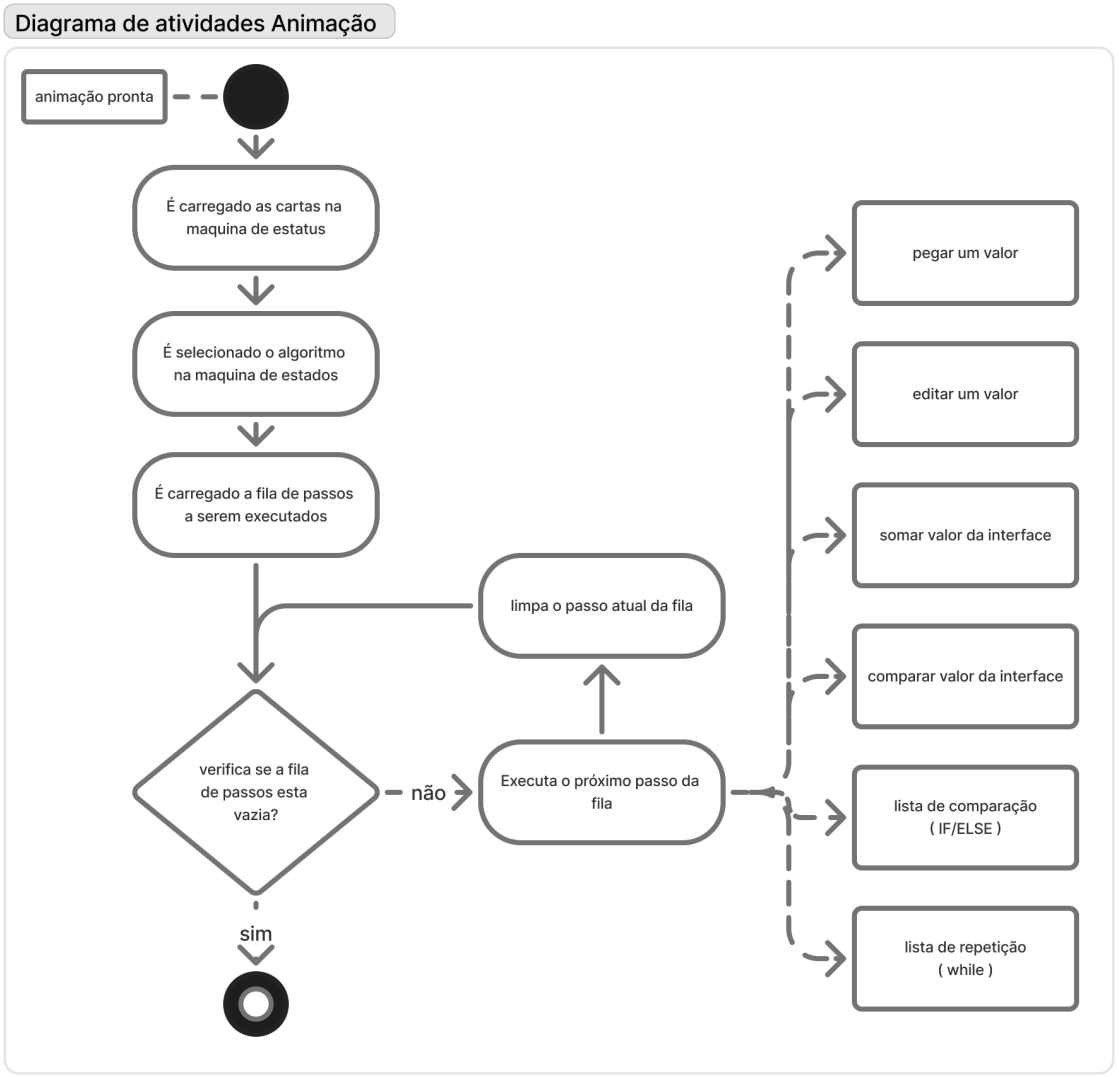
2. Vetor: 05 registradores que representam um vetor com 05 entradas.

3. Processador: 03 registradores que são usados para computar operações lógicas e matemáticas.

4. Esferas: um par de esferas que serve para demonstrar as movimentações de memórias do sistema.

A Figura 25 representa o fluxo das animações no sistema, baseado em uma máquina de estados. Essa máquina de estados é uma fila de ações executadas no modelo *first in first out* (o primeiro a chegar é o primeiro a sair), que são executadas usando os elementos da placa como entrada, memória e núcleos de processamento. Assim, tentando simular com uma certa liberdade poética como o computador faz o processamento dos algoritmos.

Figura 25: Diagrama de Atividades Animação



Fonte: Elaborado pelo autor.

As ações programadas são as seguintes:

- GetValor: função na qual é selecionada uma variável e é salvo o valor da mesma numa variável auxiliar usada como ponteiro. A função só é dada como encerrada quando uma das esferas sai do elemento selecionado até o núcleo de processamento.
- SetValor: função na qual é salvo um valor em uma variável específica, podendo ser um valor pré-definido no código do algoritmo ou um valor dinâmico, como o próprio auxiliar citado anteriormente. A função só é finalizada quando uma esfera sai do processador e chega no objeto especificado.

- SomarValor: função na qual é possível somar um valor, seja ele positivo ou negativo, de um determinado campo com um valor determinado. Isso se aplica sendo ele fixo no código ou dinâmico (auxiliar).

- CompararValor: função no qual é possível comparar os valores dos núcleos de processamento. Essa função não possui entradas, acessando diretamente os valores do processador. Os objetos do núcleo de processamento com posições 0 e 2 são os operandos e a posição 1 é a operação.

- if/Else: a função *if and else* têm como entrada duas listas de funções e uma função de comparação (CompararValor). Assim, uma vez que se obtém o valor da função de comparação, é executado internamente as funções, a função só finaliza assim que todas as ações estão finalizadas.

- While: a função *while* tem como entrada uma lista de funções e uma função de comparação (Comparar Valor). Ao iniciar a função, é executada a comparação. Se a função retorna positivo, são executadas todas as funções da lista de funções até que a mesma esteja vazia, então é executada a função de comparação novamente. Caso continue retornando verdadeiro, o ciclo se repete. É necessário salientar que é preciso recolocar os valores da comparação de volta no processador como passos finais da lista.

Falando em termos de possibilidades, essa máquina de estados atualmente é capaz de executar uma quantidade limitada de algoritmos, o que a classifica como um autômato de estados com registradores devido aos seguintes fatores.

Ainda é limitada, alcançando as operações já que ela possui limitações em termos de memória, uma vez que os registradores estão limitados por valores de [0 … 6] por motivos estéticos. Já que a exibição deles é feita usando as cartas criadas anteriormente, o que não impede futuramente a criação de outras cartas ou usar algum outro mecanismo de exibição.

Outra questão envolvendo a memória é a questão do limite de memória imposto pelo número finito de registradores, o que impede de se criar programas muito complexos, por exemplo, um algoritmo de TreeSort ou receber entradas mais extensas. Contudo, esse baixo número de registradores e entradas tem o intuito de facilitar o entendimento dos estudantes, principalmente os que estão começando a programar. Isso não é tão restritivo, haja vista que exemplos iniciais possuem normalmente entradas menores e são mais didáticos.

4.AVALIAÇÃO E RESULTADOS OBTIDOS

Nesta seção é apresentada a metodologia utilizada para avaliar o uso de realidade aumentada e visualização de algoritmos por meio da ferramenta CardSortRA como ferramenta de apoio no ensino de algoritmos.

4.1 Aplicação em Sala de Aula

O aplicativo CardSortRA foi apresentado a uma turma de calouros do curso de Ciência da Computação da Universidade Federal do Maranhão (UFMA) visando avaliar sua eficácia. O teste em sala de aula foi estruturado em três etapas: na primeira foram coletados os dados da percepção dos alunos sobre seus conhecimentos em algoritmos por meio de um formulário; em seguida, os alunos foram divididos em pequenos grupos no qual foi entregue um conjunto de cartas; e, após uma breve explicação aos alunos, eles ficaram livres para testar o aplicativo. Após esse momento, foi introduzida a segunda parte do formulário, cujo objetivo era medir o engajamento dos alunos após o uso do aplicativo e para compreender se houve uma melhora na confiança dos mesmos em seus conhecimentos sobre algoritmos.

4.1.1 Método de Avaliação

Para avaliar a confiança dos alunos em seus conhecimentos e o engajamento, utilizou-se um formulário baseado na Escala de Likert, que consiste em um método que permite medir percepções subjetivas de forma padronizada. As questões foram estruturadas em tons afirmativos, e os entrevistados deveriam avaliar o grau de concordância em uma escala de 1 a 5, onde 1 corresponde a “discordo totalmente” e 5 a “concordo totalmente”.

4.1.2 Aplicação do formulário

No total, foram feitas 18 perguntas nos formulários, divididas em sessões:

1. Identificação e familiaridade com a área com 02 questões.
2. Autoavaliação dos conhecimentos como programador com 05 questões,
3. Momento para o teste do aplicativo.
4. Momento para o teste do aplicativo, avaliação da ferramenta e seus impactos com 11.

4.1.2.1 Primeira seção do formulário

Na primeira seção, o foco reside em identificar os alunos e seu nível de experiência em relação à programação. Foram feitas duas perguntas sendo uma de identificação e uma quantitativa para saber a quanto tempo a pessoa de experiência com programação. Para a segunda pergunta, há as seguintes possíveis respostas:

- Nunca programei.
- Programa a menos de 06 meses.
- Entre 06 meses a 02 anos.
- Há mais de 02 anos.

4.1.2.2 Segunda seção do formulário

A segunda seção focou na auto avaliação dos conhecimentos como programador, por meio da análise das seguintes afirmações, conforme descrito anteriormente.

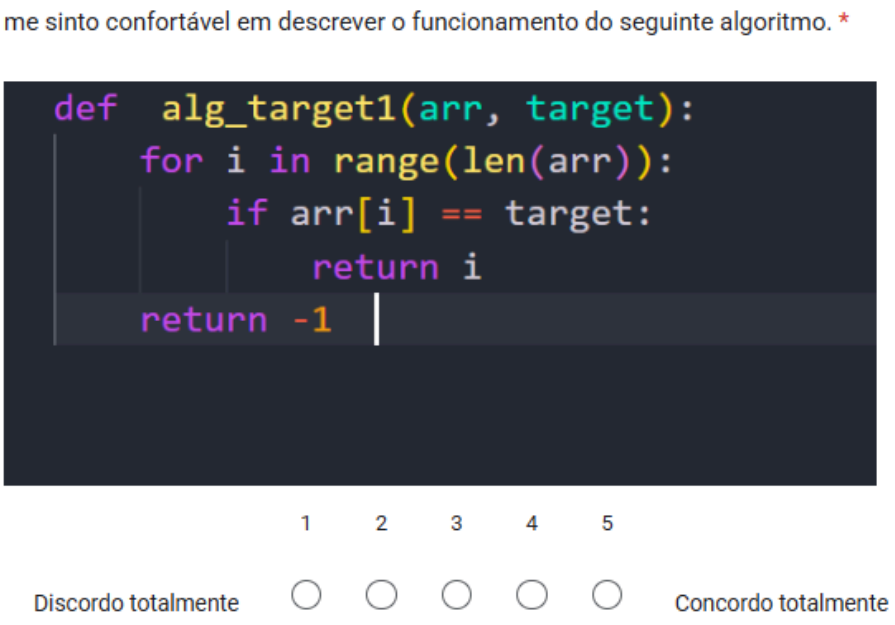
Tabela 2: Auto avaliação dos conhecimentos do programador

2. Tenho confiança nos meus conhecimentos nos conceitos de alocação de variáveis e substituição de valores.
3. Tenho confiança nos meus conhecimentos no conceito de Estrutura Condicional **(IF/ELSE).**
4. Tenho confiança nos meus conhecimentos no conceito de Laços de Repetição (WHILE/FOR)

Fonte: Elaborado pelo autor

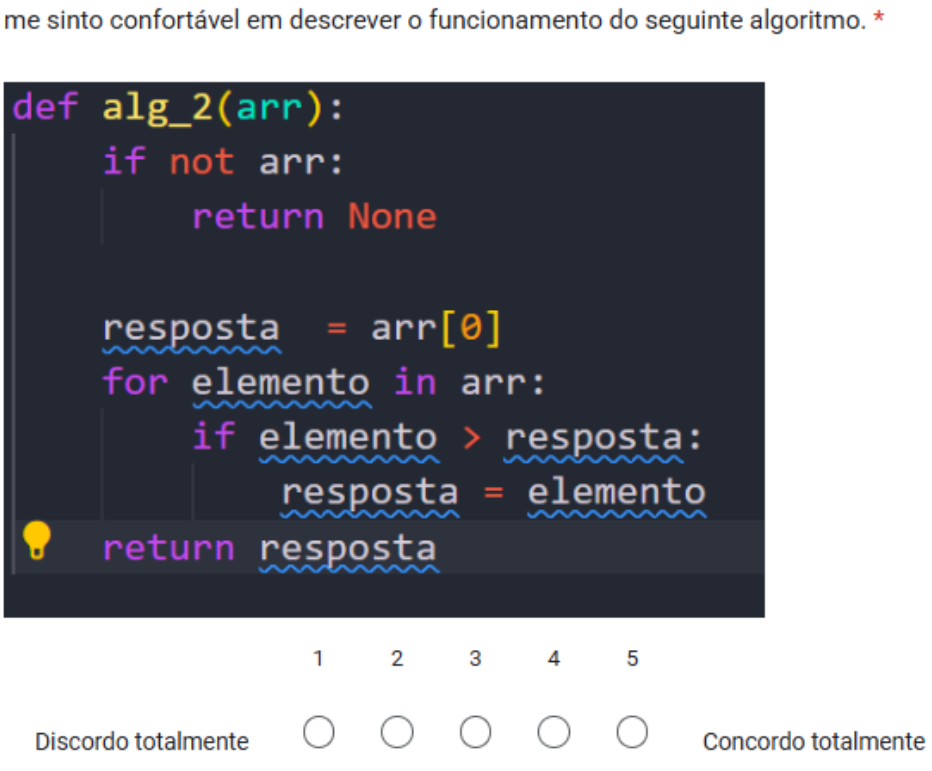
Para avaliar a capacidade de utilizar essas três competências em conjunto, foram apresentados dois algoritmos nas Figuras 26 e 27. Nas respectivas perguntas 5 e 6, solicitou-se aos respondentes que avaliassem sua própria habilidade para descrever o funcionamento desses algoritmos. Por questões metodológicas, os nomes dos algoritmos foram omitidos.

Figura 26: Busca Linear



Fonte: Elaborado pelo autor.

Figura 27: Busca pelo Maior Elemento



Fonte: Elaborado pelo autor.

4.1.2.3 Terceira seção do formulário

Após os estudantes terem completado a segunda seção do formulário, foi liberado o link para os alunos baixarem o aplicativo e se dividirem em pequenos grupos para realizar os testes. Durante vinte minutos, os alunos utilizaram o aplicativo de forma livre tendo apenas interferências para tirar dúvidas e resolver alguns eventuais problemas na aplicação.

4.1.2.4 Quarta seção do formulário

Para avaliar a capacidade de utilizar essas três competências em conjunto, foram apresentados dois algoritmos nas Figuras 26 e 27. Nas respectivas perguntas 5 e 6, solicitou-se aos respondentes que avaliassem sua própria habilidade para descrever o funcionamento desses algoritmos. Por questões metodológicas, os nomes dos algoritmos foram omitidos.

Tabela 3: Afirmações usadas na quarta seção

7. O aplicativo manteve minha atenção durante o aprendizado dos algoritmos.
8. Tive vontade de continuar usando o aplicativo após o primeiro uso.
9. As atividades do app me motivaram a tentar resolver problemas por conta própria.
10. Senti vontade de usar as cartas para resolver eu mesmo os problemas propostos.
11. Foi fácil compreender os passos dos algoritmos com a animação apresentada.
12. Senti que aprendi mais rápido com a ajuda das animações.
13. Percebi meus colegas mais animados após o uso do aplicativo.
14. Percebi meus colegas comentando mais sobre o assunto da aula após o uso do aplicativo.

Fonte: Elaborado pelo autor

Conforme mencionado anteriormente, as perguntas 15 e 16 repetem as perguntas 5 e 6, solicitando aos respondentes que indiquem com que nível de confiança poderiam identificar e descrever o funcionamento dos algoritmos apresentados nas Figuras 26 e 27. O objetivo era avaliar se houve melhora na segurança dos alunos em descrever os algoritmos de busca linear e busca pelo maior elemento. Ao final, foi solicitado que fornecessem sugestões de melhorias.

4.2 AVALIAÇÃO DOS RESULTADOS

O formulário de avaliação do CardSortRA contou com um total de 23 respondentes. Desses participantes, a maioria (82%) possuía menos de seis meses de experiência em programação, sendo 4,3% indivíduos que nunca haviam programado e 78,3% com até seis meses de experiência. Quanto aos demais, 13% dos respondentes tinham experiência entre seis meses e dois anos de programação, enquanto 4,3% possuíam mais de dois anos de experiência na área.

Figura 28: Gráfico de experiência dos respondentes

Fonte: Elaborado pelo autor

4.2.1 Aprofundando no nível de conhecimento dos entrevistados.

Indo mais em detalhes nas áreas específicas dentro dos conceitos de programação e, conforme os dados coletados pelo formulário, 78,26% das respostas declaram que os participantes têm confiança total ou parcial nos seus conhecimentos de declaração de variável, tendo 21,73% como neutras e zero negativa.

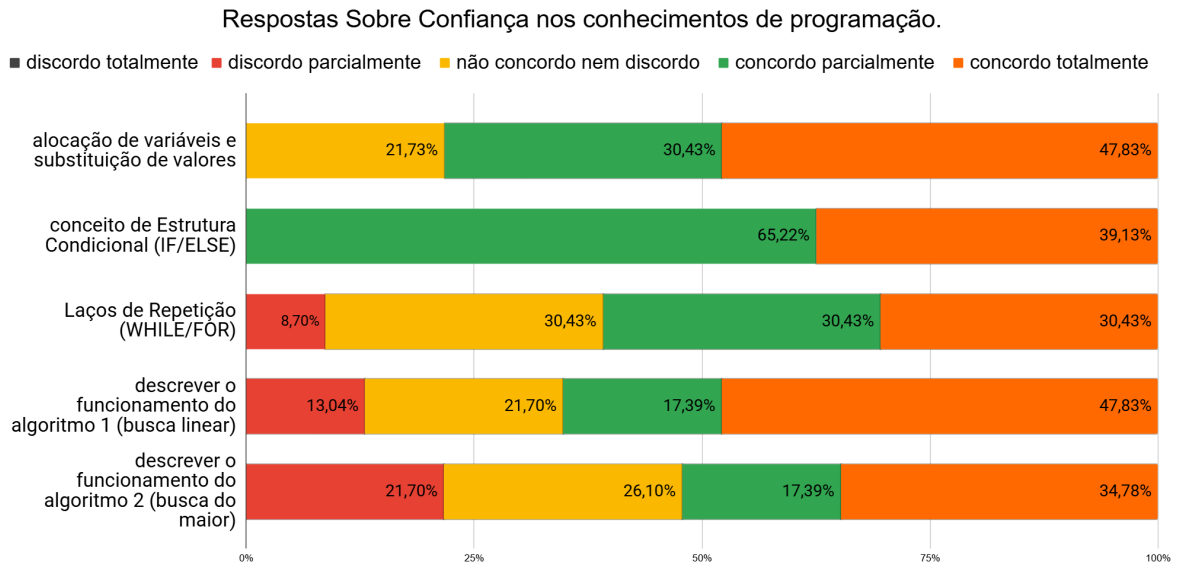
Já quando falamos dos conceitos de estruturas condicionais, o que se é possível observar na amostra é que dentro das áreas questionadas, esta é a área em que a turma mais se sente confiante, já que todos os entrevistados afirmaram ter confiança parcial ou total nos seus conhecimentos, sendo 65,22% parcial e 39,13% como confiança total.

Em contrapartida, a área de laços de repetição é o que, dentro da amostra, os alunos afirmaram se sentir mais inseguros. Uma vez que quase 9% dos alunos não se sentiam seguros nos seus conhecimentos de laços de repetição, 30% afirmam que não se sentem nem seguros, nem inseguros e menos de 2/3 da turma afirmaram que se sentem confiantes com o assunto, com 30% afirmado que se sente confiante parcialmente e 30% confiante totalmente nesse conceito de programação.

Enquanto a confiança em descrever o funcionamento dos algoritmos de busca linear e busca pelo maior elemento mostram resultados parcialmente positivos tendo os respondentes afirmando que se sentiam confiantes para descrever o funcionamento dos algoritmos parcial ou totalmente com mais de 50% sendo respectivamente:

Busca linear com 17,4% se sentido parcialmente confiante e 47,8% se sentido totalmente confiante, 21% neutro e 13% sentido parcialmente inseguro. Busca pelo maior elemento, 17,4% parcialmente confiante em descrever o funcionamento, 34,7% totalmente confiante, 26,1% neutro e 21,7% parcialmente inseguro.

Figura 29: Gráfico de respostas sobre confiança nos conhecimentos em programação



Fonte: Elaborado pelo autor

4.2.2 Avaliação do aplicativo

4.2.2.1 Engajamento

Em relação à experiência com o aplicativo, constata-se que, segundo as respostas obtidas, o aplicativo alcançou um nível satisfatório de engajamento para um MVP (Mínimo Produto Viável). Mais de 80% dos respondentes afirmaram que a ferramenta conseguiu manter sua atenção durante o uso, sendo 47% de concordância total e 39% de concordância parcial com a afirmação 'O aplicativo manteve minha atenção durante o aprendizado dos algoritmos'. Os demais respondentes se distribuíram entre neutros (9%) e discordantes (4,3%).

4.2.2.2 Reusabilidade

Quanto à reusabilidade do aplicativo, este foi um dos aspectos com desempenho menos satisfatório e que demanda melhorias. A análise das respostas à afirmação 'Tive vontade de continuar usando o aplicativo após o primeiro uso' revelou que quase 39% dos entrevistados reagiram de forma neutra ou negativa, com a seguinte distribuição: 4% discordaram totalmente, 30% não concordaram nem discordaram, 26% concordaram parcialmente e 35% concordaram totalmente.

Vale destacar que, entre as sugestões de melhorias, as mais frequentes referem-se à interface do aplicativo, fator que pode ter influenciado negativamente sua reusabilidade.

4.2.2.3 Clareza nas explicações

Para avaliar o impacto educacional direto da ferramenta, utilizaram-se duas perguntas. A primeira - 'Foi fácil compreender os passos dos algoritmos com as animações apresentadas?' - Visava mensurar a clareza das explicações animadas. Os resultados obtidos foram parcialmente neutros e não tão positivos quanto o esperado: aproximadamente 40% das avaliações foram neutras ou negativas (4% discordaram totalmente, 13% discordaram parcialmente e 22% permaneceram neutros), enquanto 61% concordaram parcial ou totalmente.

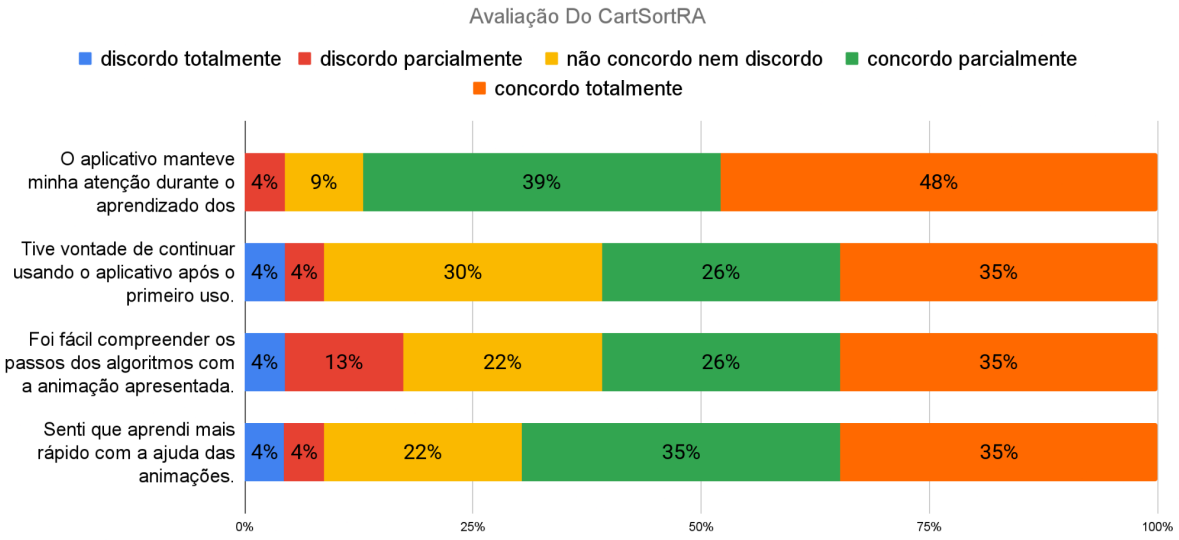
Esses dados corroboram uma das principais sugestões de melhoria apontadas pelos usuários: a necessidade de um tutorial ou de explicações mais claras no aplicativo.

4.2.2.4 Percepção no aprendizado

Para mensurar a percepção dos alunos sobre a eficácia do aplicativo no processo de aprendizagem, utilizou-se a afirmação: 'Senti que aprendi mais rápido com a ajuda das animações'. Diferentemente das respostas anteriores, a avaliação dessa afirmação apresentou 70% de concordância (parcial ou total), distribuídos igualmente em 35% para cada opção, além de 22% de respostas neutras, 4% de discordância parcial e 4% de discordância total.

Uma hipótese para essa melhoria de 10% nas avaliações, em comparação com as perguntas anteriores, pode estar relacionada aos impactos indiretos do aplicativo em sala de aula.

Figura 30: Gráfico de respostas sobre avaliação do CardSortRA



Fonte: Elaborado pelo autor.

4.2.2.5 Sugestões e feedbacks

A coleta das sugestões e dos feedbacks foi feita por meio de um campo de texto onde o respondente podia deixar por escrito qualquer informação ou feedback que ele achasse relevante a ser melhorado ou corrigido. Entre os feedbacks, os que mais se destacaram foram bugs relacionados a erros de leitura, com cinco respostas mencionando esse problema. Os estudantes destacaram que as cartas de busca pelo maior e de busca pelo menor, por serem muito parecidas, eram apenas lidas como busca pelo maior. Além disso, também foi constatado que, durante a aplicação, alguns celulares tinham mais dificuldade de identificar as

cartas ou as superfícies, assim atrapalhando a experiência do aplicativo. Ademais, outro ponto dentro desse tópico é que foi descoberto que a biblioteca da Vuforia apresenta dificuldades em lidar com vários objetos ao mesmo tempo, o que acaba acarretando erros e bugs.

Este feedback condiz com o segundo ponto mais mencionado no formulário, que seria a questão da compatibilidade do aplicativo, que foi testado apenas em alguns celulares Android devido à incompatibilidade de certas versões do Android com a versão da Unity utilizada durante o desenvolvimento. Ademais, não foi disponibilizada uma versão para celulares iOS, uma vez que para fazer a distribuição dos aplicativos, mesmo que por fora da loja oficial da Apple, se faz necessário uma conta de desenvolvedor da Apple que tem custo médio de $ 100 dólares, assim tornado proibitiva a distribuição do protótipo na plataforma iOS.

Tabela 4: Sugestões e feedbacks

Feedbacks envolvendo erros de escaneamento (RA)	5
Feedbacks envolvendo compatibilidade	3
Feedbacks envolvendo erros e bugs não categorizados	2
FeedBacks envolvendo animação confusa ou erros de animação	2
FeedBacks envolvendo melhorias na interface	2
FeedBacks envolvendo a proposta do aplicativo	1

Fonte: Elaborado pelo autor

4.2.3 Avaliação dos impactos do aplicativo no contexto de uma sala de aula

Como o aplicativo educacional em questão foi desenvolvido para complementar as atividades em sala de aula, faz-se necessário avaliar igualmente seu impacto no ambiente educacional. Nesse contexto, a experiência com o CardSortRA demonstrou resultados positivos, especialmente no que diz respeito à motivação, ao estímulo da interação e à troca de conhecimentos entre os participantes.

Para mensurar o nível de motivação dos alunos após a utilização do aplicativo, adotaram-se as seguintes afirmações:

1. “As atividades do app me motivaram a tentar resolver os problemas propostos por conta própria”.
2. “Senti vontade de usar as cartas para resolver eu mesmo os problemas propostos”.

Em relação à primeira afirmação, observou-se uma resposta majoritariamente positiva, com 39% concordando totalmente e 30% concordando parcialmente, totalizando 69%. Já 26% dos respondentes não discordaram nem concordaram e 4,4% discordaram totalmente, indicando que dentro do grupo perguntado pode-se observar um aumento na motivação para resolver problemas de programação após o uso do aplicativo proposto.

Quanto à segunda afirmação, os resultados foram ainda mais expressivos: 52% dos alunos concordaram totalmente e 13% concordaram parcialmente, resultando em 65% de concordância. Os participantes neutros representaram 21%, enquanto 12% manifestaram discordância, seja parcial ou total. Esses dados dentro da amostra sugerem que o aplicativo teve um impacto positivo em motivar os estudantes na resolução de problemas, principalmente pelo elemento lúdico das cartas.

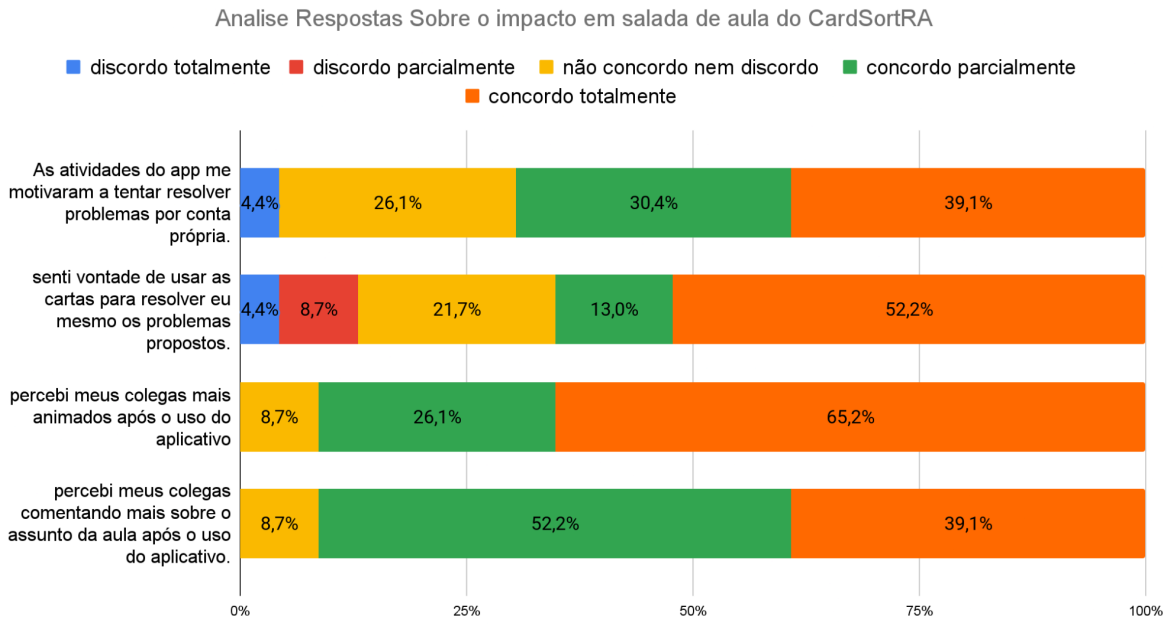
Em relação a incentivar a troca de informações entre os estudantes, esperava-se que, somado à experiência do aplicativo, mais o fato dos alunos estarem divididos em pequenos grupos, pudessem influenciar positivamente a experiência dos alunos. Para medir esse fator, foram usadas as seguintes afirmações:

1. Percebi meus colegas mais animados após o uso do aplicativo.
2. Percebi meus colegas comentando mais sobre o assunto da aula após o uso do aplicativo.

Sobre a primeira afirmação pode se observar que a visão dos estudantes em relação à motivação da turma após o uso do aplicativo foi completamente positiva, tendo 65% dos respondentes concordando totalmente com a afirmação, enquanto 26% concordando parcialmente, e somente 8,7% dos estudantes respondendo como indiferente e não houve discordância sobre essa afirmação.

Já sobre os resultados obtidos pela segunda afirmação apontaram para um sentimento de melhoria, ao menos parcial, em relação ao quanto os alunos interagiram sobre o assunto da matéria, tendo 39% dos respondentes concordando totalmente com essa afirmação e 52% respondendo que concordam parcialmente tendo aproximadamente 9% respondendo de forma neutra.

Figura 31: Gráfico dos impactos em sala de aula do CardSortRA



Fonte: Elaborado pelo autor

4.2.4 Comparação na percepção de confiança

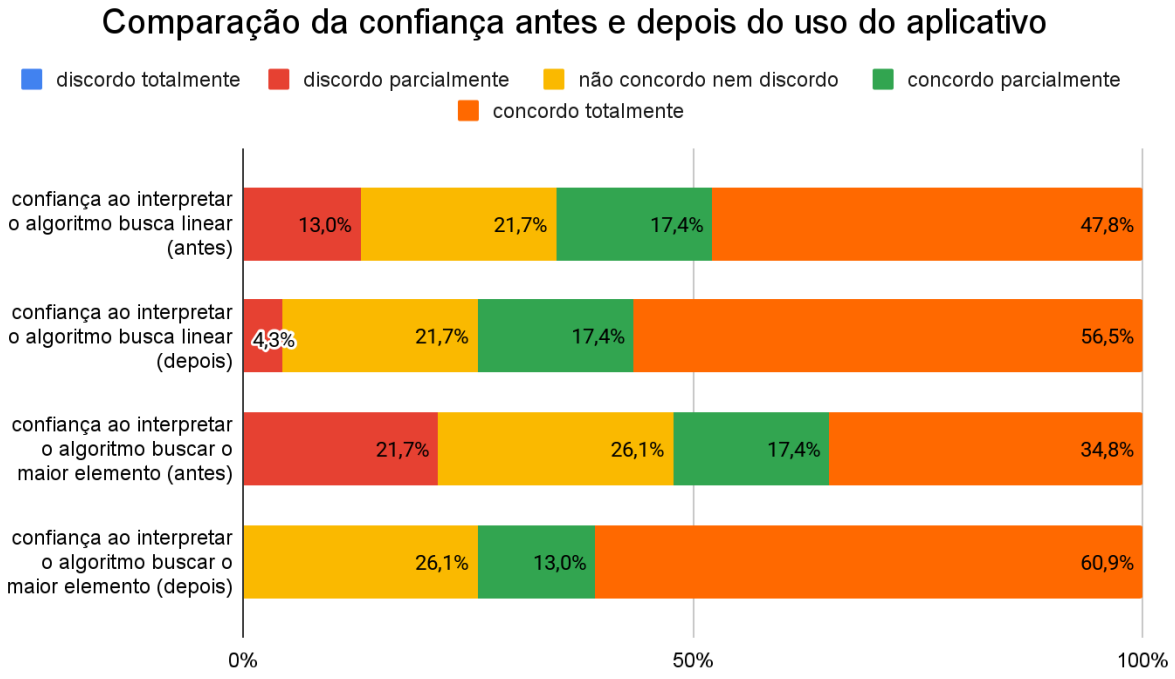
Como ponto de referência para medir se o aplicativo teve algum impacto nos alunos, foi solicitado que eles avaliassem o quão confiante se sentiam para descrever o funcionamento dos algoritmos de busca linear e de busca pelo maior elemento antes e após o uso do aplicativo.

Em relação à evolução na confiança para interpretar o algoritmo de busca linear, observou-se uma melhora sutil. A porcentagem de respondentes que se sentiam parcialmente inseguros caiu de 13% para apenas 4,3%, enquanto aqueles que se sentiram totalmente seguros aumentaram de 47,8% para 56,5%. Os demais valores permaneceram inalterados: 0% discordaram totalmente, 21,7% mantiveram-se neutros e 17,4% concordaram parcialmente.

Quanto à avaliação da confiança em descrever o algoritmo de busca pelo maior elemento, o aumento foi mais significativo. O percentual de alunos que se declararam parcialmente inseguros caiu de 21,7% para 0%, ou seja, nenhuma resposta indicou qualquer tipo de insegurança. Os que responderam de forma neutra permaneceram em 26,1%, enquanto os parcialmente seguros diminuíram levemente, de 17,4% para 13%. Por outro lado, o grupo que se declarou totalmente confiante cresceu de forma expressiva, passando de 34,8% para 60,9%.

No entanto, apesar dessas melhorias, é necessário fazer algumas ressalvas. O aumento na confiança não implica, necessariamente, uma melhora nas habilidades como programador, e tampouco avaliar isso é o objetivo central desta pesquisa.

Figura 32: Gráfico de Comparação da confiança antes e depois do uso do aplicativo



Fonte: Elaborado pelo autor

5. CONSIDERAÇÕES FINAIS

Conforme as respostas do questionário sobre o teste em sala de aula do CardSortRa, após o uso do aplicativo, houve indícios de melhora no engajamento e na confiança dos alunos que realizaram os testes. Esses resultados nos levam a crer que há indícios de um potencial educacional para o uso de realidade aumentada e visualização como ferramenta de auxílio para gerar engajamento e motivação dos estudantes de programação. Porém, para se ter uma hipótese sólida seria necessária realização de testes mais elaborados e sistemáticos e com uma amostra maior, sem contar no uso de ferramentas de medição que consigam medir outros pontos-chave como, por exemplo, se houve uma real melhoria no aprendizado dos estudantes e a avaliação da ferramenta pelos professores.

Outro ponto importante a se mencionar é que, por mais que as tecnologias para o desenvolvimento de experiências em realidade aumentada tenham se tornado cada vez mais acessíveis, ainda se têm muitos pontos a serem melhorados, principalmente quando falamos da criação de projetos mobile mais complexos. Há problemas de compatibilidade de hardware e software, como presenciado nos testes em sala de aula, uma vez que a maioria das câmeras de celular ainda não possui tecnologias para medição precisa de distância, o que acarreta problemas na hora da leitura e projeção das animações. Outro fator que pode acabar sendo desafiador são as limitações das *engines* como a Unity e Vuforia, uma vez que apresentaram limitações em relação a aspectos como o *multitracking* e cálculo de posição global de objetos. Como citado anteriormente, esse foi o motivo pelo qual se fez necessária a criação de um ponto de referência físico para capturar a ordem das cartas. Esses fatores não são impeditivos, mas podem ser desafiadores ou acarretar uma perda de qualidade da experiência do usuário.

Mesmo com esses desafios a serem superados, o futuro da realidade aumentada dentro do contexto educacional apresenta-se ser promissor. Como próximos passos, o planejado é continuar melhorando o aplicativo, com a correção dos bugs apontados pelos alunos durante os testes e transformar a atual máquina de estados que executa as animações em um compilador funcional. Isso permite aos professores e estudantes executarem os próprios códigos dentro do aplicativo. Sem contar a oportunidade de estar desenvolvendo outros projetos similares para testar os mesmos critérios de compatibilidade com dispositivos e recursos em outras *engines* como, por exemplo, a recém-lançada *Unreal Engine* que tem apresentado projetos promissores, mas ainda com poucas menções a ela em pesquisas e artigos.

REFERÊNCIAS

AZUMA, Ronald T.; BAILLOT, Yohan; BEHRINGER, Reinhold; FEINER, Steven; JULIER, Simon; MACINTYRE, Blair. **Recent advances in augmented reality**. IEEE Computer Graphics and Applications, v. 21, n. 6, p. 34–47, nov./dez. 2001. DOI: 10.1109/38.963459.

Astute Analytica India PVT. LTD. **Global Game Engine Market to Worth Over US$ 12.84 Billion by 2033**. GlobeNewswire [S.l.], 15 abr. 2025. Disponível em: https://www.globenewswire.com/news-release/2025/04/15/3061731/0/en/Global-Game-Engine-Market-to-Worth-Over-US-12-84-Billion-By-2033-Astute-Analytica.html. Acesso em: 11 jul. 2025.

BAECKER, Ronald. Sorting out sorting: a case study of software visualization for teaching computer science. In: STASKO, John T.; DOMINGUE, John; BROWN, Marc H.; PRICE, Blaine A. (Org.). **Software visualization**: programming as a multimedia experience. Cambridge: MIT Press, 1998. p. 369–381.

Haas, John. **A History of the Unity Game Engine**. Trabalho de conclusão de curso (Interactive Qualifying Project) – Worcester Polytechnic Institute, Worcester, M Massachusetts, 2025.

Holfeld, Julian. **On the relevance of the Godot Engine in the indie game development industry**. arXiv, 27 dez. 2023. DOI: 10.48550/arXiv.2401.01909. Disponível em: https://arxiv.org/abs/2401.01909. Acesso em: 11 jul. 2025.

PTC INC. **Vuforia Library**. Vuforia Developer Portal, [s. d.]. Disponível em: https://developer.vuforia.com/library/. Acesso em: 11 jul. 2025.

_____. **Vuforia Engine**: Augmented Reality SDK. Disponível em: https://developer.vuforia.com/. Acesso em: 01 ago. 2025.

Simon, János. **Augmented reality application development using Unity and Vuforia**. INDECS – Interdisciplinary Description of Complex Systems, Szeged, v. 21, n. 1, p. 61–69, 2023. DOI: 10.7906/indecs.21.1.6. Disponível em: https://indecs.eu/2023/indecs2023-pp61-69.pdf. Acesso em: 11 jul. 2025.

_____. **Augmented reality application development using Unity and Vuforia**. Interdisciplinary Description of Complex Systems, v. 21, n. 1, 2023. Disponível em: https://www.researchgate.net/publication/371462889_Augmented_Reality_Application_Development_using_Unity_and_Vuforia. Acesso em: 20 jun. 2025.

Tori, R., Kirner, C e Siscoutto, R. **"Fundamentos e tecnologias de realidade virtual e aumentada"**. Porto Alegre: Sociedade Brasileira de Computação, 2006. 422p.

Unity Technologies. **XR Glossary**. Unity, [s. d.]. Disponível em: https://unity.com/how-to/xr-glossary. Acesso em: 11 jul. 2025.

_____. **GameObjects**. Unity Manual, [s. d.]. Disponível em: https://docs.unity3d.com/Manual/GameObjects.html. Acesso em: 11 jul. 2025.

Vieira, Newton José. *Máquinas e Estados Finitos*. IN: Vieira, Newton José. **Introdução aos fundamentos da computação**. Belo Horizonte: UFMG, [s.d.]. Disponível em: https://homepages.dcc.ufmg.br/~nvieira/cursos/tl/a17s2/livro/cap2.pdf#page=9. Acesso em: 11 jul. 2025.

Yamashita, Koichi; Fujioka, Ryota; Kogure, Satoru; Noguchi, Yasuhiro; Konishi, Tatsuhiro; Itoh, Yukihiro. **Practices of algorithm education based on discovery learning using a program visualization system**. Research and Practice in Technology Enhanced Learning, [S.l.], v. 11, n. 15, 2016.