CIÊNCIA& TECNOLOGIA
ISSN: 2525-6599
unifev pesquisa
Coordenação de Pesquisa

AUXÍLIO EDUCACIONAL PARA CRIANÇAS COM TEA POR MEIO DA GAMIFICAÇÃO

Miguel Santos Oliveira[1]
Fernando Bermejo Menechelli[2]

Resumo:
O Transtorno do Espectro Autista (TEA) é uma condição do neurodesenvolvimento caracterizada por alterações na comunicação, na interação social e no comportamento. Nesse contexto, este artigo apresenta o desenvolvimento do jogo digital Funbetização, concebido para auxiliar o processo de alfabetização de crianças com TEA por meio da associação entre letras, palavras e imagens do cotidiano. A proposta surgiu a partir da observação de um aluno não verbal do Colégio Unifev, situado em Votuporanga, estado de São Paulo, que utiliza um tablet como principal meio de comunicação, evidenciando a importância de recursos tecnológicos acessíveis e adaptados às necessidades desse público. O jogo foi desenvolvido com fases que utilizam reforços positivos imediatos, controle do ritmo de aprendizagem e suporte visual, respeitando as particularidades sensoriais dos usuários. A metodologia adotada baseou-se na análise orientada a objetos, utilizando a modelagem UML, enquanto o desenvolvimento empregou a plataforma Unity, a linguagem C# e a exportação para o sistema Android. Os testes realizados indicaram que o Funbetização constitui um recurso eficaz de apoio à alfabetização e à inclusão de crianças com TEA, favorecendo a atenção, o engajamento e a aprendizagem por meio de pistas visuais, repetição estruturada e elementos de gamificação. Conclui-se que o jogo atingiu seu objetivo de proporcionar uma experiência interativa, educativa e acessível, demonstrando o potencial da gamificação como estratégia para tornar o processo de alfabetização mais significativo, lúdico e motivador para crianças autistas.

Palavras-chave: gamificação; TEA; jogo educativo; inclusão; alfabetização.

Abstract:
Autism Spectrum Disorder (ASD) is a neurodevelopmental condition characterized by impairments in communication, social interaction, and behavior. In this context, this article presents the development of the digital game Funbetização, designed to support the literacy process of children with ASD through the association of letters, words, and everyday images. The proposal originated from the observation of a nonverbal student at Colégio Unifev, located in the city of Votuporanga, São Paulo State, Brazil, who uses a tablet as his primary means of communication, highlighting the importance of accessible technological resources tailored to the needs of this population. The game was designed with multiple stages that incorporate immediate positive reinforcement, learner-paced progression, and visual support while respecting the sensory characteristics of its users. The methodology was based on object-oriented analysis using Unified Modeling Language (UML), while the game was developed with the Unity platform, the C# programming language, and deployed for the Android operating system. The results of the tests indicated that Funbetização is an effective tool for supporting literacy and promoting the inclusion of children with ASD by enhancing attention, engagement,

[1] Discente do curso de Engenharia de Computação da Unifev - Centro Universitário de Votuporanga. Votuporanga, São Paulo, Brasil. Email: miguel.santos321_@hotmail.com.
[2] Docente do curso de Engenharia de Computação da Unifev - Centro Universitário de Votuporanga. Votuporanga, São Paulo, Brasil. Mestre em Medicina Biomédica Email: fernandomenechelli@fev.edu.br.

and learning through visual cues, structured repetition, and gamification elements. It is concluded that the game successfully achieved its objective of providing an interactive, educational, and accessible learning experience, demonstrating the potential of gamification as a strategy for making the literacy process more meaningful, engaging, and motivating for children with autism.

Keywords: gamification; ASD; educational game; inclusion; literacy.

INTRODUÇÃO

O Transtorno do Espectro Autista (TEA) é uma condição neurobiológica do desenvolvimento que compromete a comunicação, a interação social e o comportamento, apresentando diferentes níveis de suporte, conforme a intensidade dos sintomas. Segundo o Manual Diagnóstico e Estatístico de Transtornos Mentais (DSM-5), o TEA é caracterizado por déficits persistentes na comunicação e interação social em múltiplos contextos, bem como por padrões restritos e repetitivos de comportamento, interesses ou atividades (APA, 2013). A heterogeneidade do espectro significa que cada indivíduo com autismo pode apresentar combinações e intensidades distintas desses sintomas, o que torna essencial a personalização de estratégias de ensino e cuidado.

O diagnóstico do TEA é categorizado em três níveis de suporte, que variam de leve (nível 1) a severo (nível 3), conforme o grau de comprometimento das habilidades adaptativas e a necessidade de suporte especializado. Indivíduos classificados no nível 3 do TEA necessitam de suporte substancial e contínuo para realizar atividades cotidianas, especialmente em contextos educacionais (Baracho, 2024; Miliausks; Pinheiro, 2024). Esse grupo geralmente apresenta déficits severos na comunicação verbal e não verbal, além de dificuldades intensas de flexibilidade comportamental, o que pode resultar em barreiras significativas no acesso ao currículo escolar tradicional.

Dentro desse espectro, destaca-se a situação das crianças não verbais, que enfrentam desafios ainda mais profundos no processo de alfabetização. A ausência da linguagem oral não implica ausência de cognição, mas exige metodologias adaptadas que valorizem outras formas de expressão e aprendizagem. Crianças com autismo não verbal muitas vezes se comunicam por meio de tecnologias assistivas, como tablets com aplicativos de Comunicação Aumentativa e Alternativa (CAA), ou utilizam gestos, imagens e símbolos visuais para se expressarem. Por isso, é fundamental o uso de abordagens que respeitem essas particularidades, priorizando recursos visuais, tecnológicos e sensoriais.

A dificuldade na aquisição da linguagem escrita por parte dessas crianças levanta um problema central: como tornar o processo de alfabetização mais acessível, motivador e eficiente para o público autista não verbal? Diante dessa questão, este trabalho tem como objetivo o desenvolvimento e a avaliação de um jogo educacional digital, intitulado *Funbetização*, voltado à alfabetização de crianças com TEA, especialmente daquelas que utilizam recursos tecnológicos como meio principal de comunicação.

O jogo foi idealizado a partir da observação direta de um aluno não verbal matriculado no Colégio Unifev, que utiliza um tablet como principal ferramenta de expressão. A proposta central consiste em criar um ambiente interativo e lúdico que auxilie no reconhecimento das letras do alfabeto e na formação de palavras simples, associadas a imagens de objetos e animais do cotidiano, proporcionando uma aprendizagem mais prazerosa e eficaz. A ludicidade, aliada à repetição positiva e ao estímulo sensorial, tem se mostrado uma estratégia eficaz no ensino de crianças com TEA, pois respeita seu tempo de processamento, reduz a ansiedade e aumenta o engajamento com a atividade proposta.

A justificativa para o desenvolvimento do *Funbetização* reside na carência de recursos educacionais digitais acessíveis e eficazes voltados especificamente para o público com TEA. Embora haja um crescimento no uso de tecnologias na educação inclusiva, muitas das soluções disponíveis no mercado ainda não contemplam as necessidades específicas de crianças não verbais ou com déficits severos de interação. Ao empregar mecanismos de gamificação — como reforços positivos imediatos, controle de ritmo, suporte visual e estrutura de fases progressivas —, o jogo busca alinhar-se às necessidades cognitivas e sensoriais dessas crianças, promovendo uma experiência de aprendizagem mais inclusiva e engajadora.

Como metodologia, foi escolhida a análise orientada a objetos com base no padrão UML, juntamente com uma abordagem de prototipação iterativa. O desenvolvimento do jogo incluiu etapas de coleta de dados, análise das necessidades dos usuários; a parte de renderização e programação foi desenvolvida por meio da game engine Unity, utilizando a linguagem C#, com foco em dispositivos Android. O jogo possui oito fases, nas quais o aluno deve associar letras a palavras e imagens, promovendo o reconhecimento visual e sonoro de forma interativa. Testes preliminares indicam que o *Funbetização* apresenta bons resultados na manutenção da atenção, no engajamento e na evolução da leitura de palavras simples por crianças com TEA, demonstrando o potencial da tecnologia como ferramenta de apoio à educação inclusiva. Observou-se também uma melhora na autonomia do aluno durante as atividades propostas, além de maior envolvimento dos responsáveis e professores no acompanhamento do progresso dos alunos.

Assim, este projeto contribui com uma proposta concreta para o uso da gamificação no processo de alfabetização, reforçando a importância da integração entre educação, tecnologia e acessibilidade.

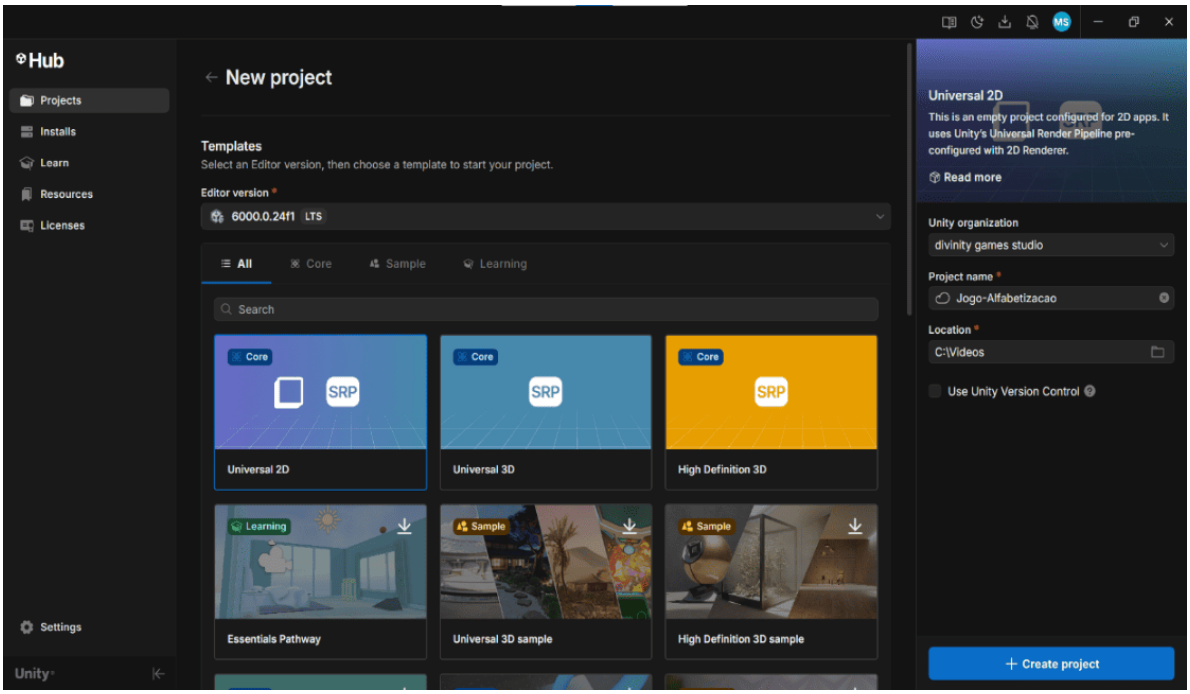
1 METODOLOGIA

Realizou-se a análise de requisitos por meio de pesquisas realizadas no Colégio Unifev, buscando identificar quais mecânicas e funcionalidades o jogo deveria possuir. A partir de entrevistas com a cuidadora do aluno com Transtorno do Espectro Autista (TEA) matriculado na instituição, foram definidas as mecânicas básicas do jogo, como o arraste das letras por meio do toque na tela do celular e a reprodução do som correspondente quando o aluno/usuário posiciona corretamente a letra no local indicado.

1.1 Criando o projeto inicial na Unity

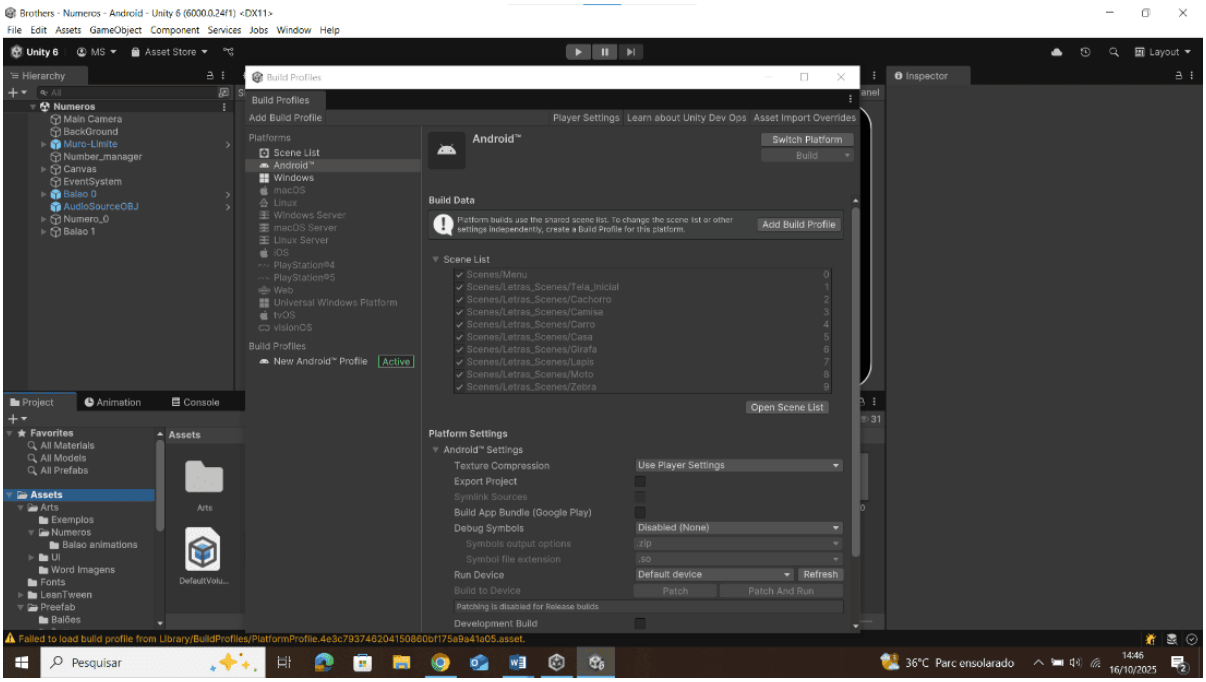
Nesta etapa, criou-se o projeto no Unity Hub, detalhando o seu nome e o diretório onde será salvo e foi configurada a sua build para rodar em dispositivos Android.

Figura 1- tela criação de projeto Unity Hub



Fonte: do autor (2025).

Figura 2-tela de configuração de build para android



Fonte: do autor (2025).

1.2 Iniciando o desenvolvimento

Após a configuração do projeto e a definição da plataforma em que ele será exportado, deu-se início ao desenvolvimento do protótipo. Antes de implementar a arte ou qualquer outro tipo de elemento visual, foi programada a funcionalidade básica de tocar e arrastar, utilizando a biblioteca de Input da Unity. A funcionalidade de toque e de reconhecer comandos identifica o clique do mouse como o toque único na tela touch do dispositivo. Para reconhecer o deslizar do dedo na tela e atribuir o posicionamento da letra em relação ao dedo do usuário, foi utilizada a função OnMouseDrag(). Dentro dela função foram passadas as coordenadas da posição do dedo em relação ao mundo do jogo. O Script ficou desta maneira.

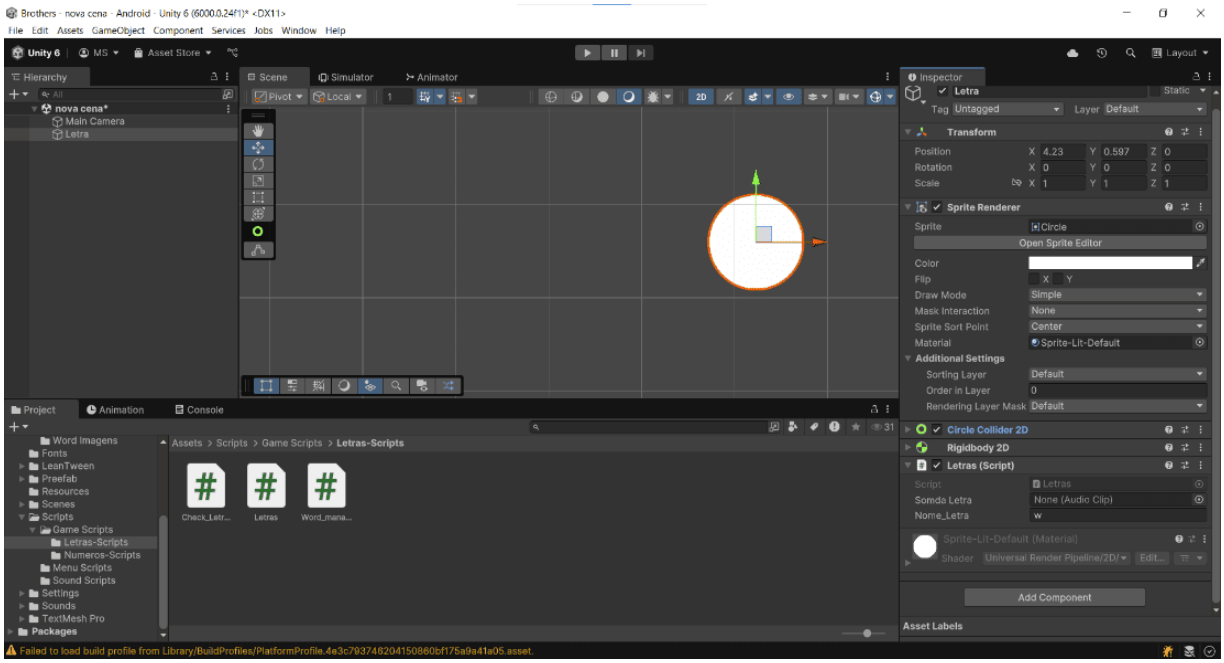
```
private void OnMouseDrag()
    {
        Vector2 dir = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        transform.position = dir;
    }
```

O Vector2 armazena as coordenadas X e Y de um objeto durante a execução do jogo. Nesse comando, obtém-se a posição do dedo na tela e converte-se essa posição em uma

coordenada correspondente ao mundo do jogo. Dessa forma, enquanto o objeto possuir componentes de física e colisão, o usuário poderá interagir com esse elemento por meio do toque na tela.

A função OnMouseDrag()**,** disponibilizada pela classe MonoBehaviour da Unity, é responsável por permitir a movimentação de objetos durante a interação do usuário. Da mesma forma, os componentes de física e colisão constituem recursos nativos (*features*) presentes na engine. Assim, ao aplicar esse *script* a um objeto da cena, torna-se possível movê-lo por meio do toque na tela de dispositivos móveis.

Figura 3 - aplicação de Script no protótipo da letra.



Fonte: do autor.

Posteriormente à implementação da mecânica básica de clicar e arrastar as letras, foi necessário adicionar novas variáveis ao script *d*a letra para possibilitar o reconhecimento e a distinção de cada uma delas, bem como o armazenamento do clipe de áudio que será reproduzido quando o aluno posicionar corretamente a letra. Como parte da expansão desse *script*, foram criadas quatro variáveis, descritas a seguir:

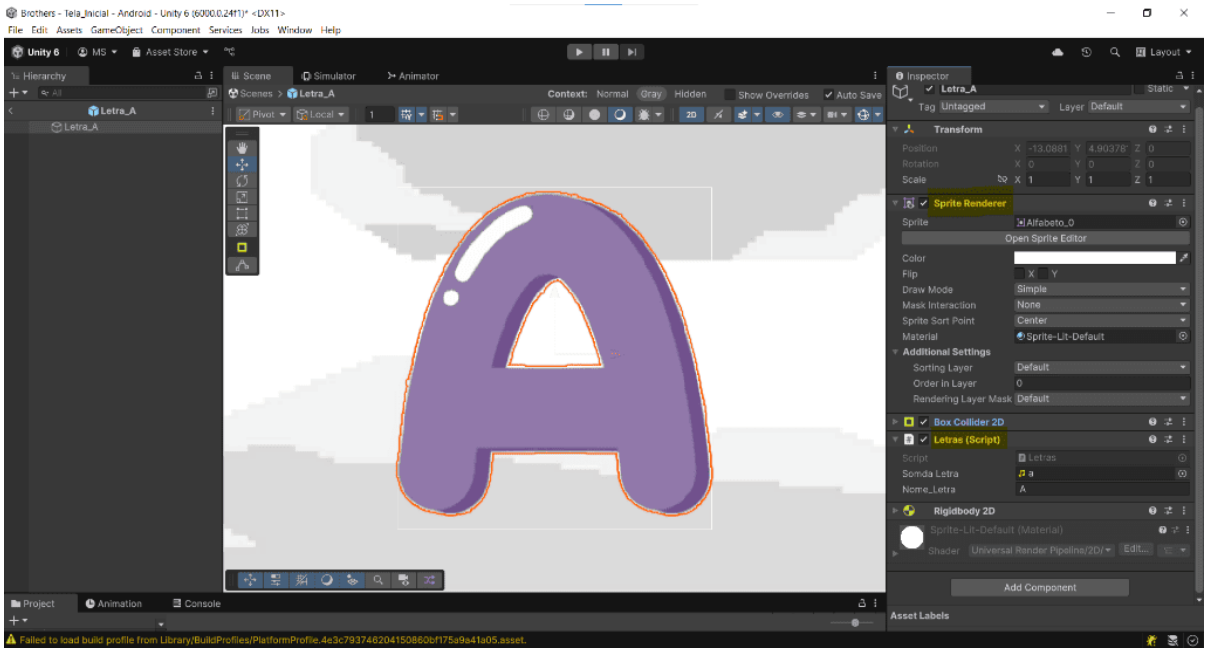
```
public AudioClip SomdaLetra;
private AudioControl AC;
public char Nome_Letra;
private Rigidbody2D rb;
```

O AudioClip vai armazenar o som da letra, o AudioControl vai tocar esse áudio, a variável char serve para dar nome à letra e ser um identificador dentro do código e sistema de colisão e por fim, o RigidBody2D, classe que comanda a física do objeto dentro do código. Para finalizar o código das letras, foi preciso criar duas funções:


```
void Update()
                {
                   if (rb.linearVelocity.y <1f)
                   {
                      rb.linearVelocity = Vector2.zero;
                   }
                }
                public void SomLetra()
                {
                   AC.Tocar_SFX(SomdaLetra);
                }
```

Com o código finalizado, foi possível realizar testes com vários objetos em cena para ver se nenhuma letra se entrelaça na colisão uma a outra, evitando, assim, que trave a sua movimentação no espaço de tela.

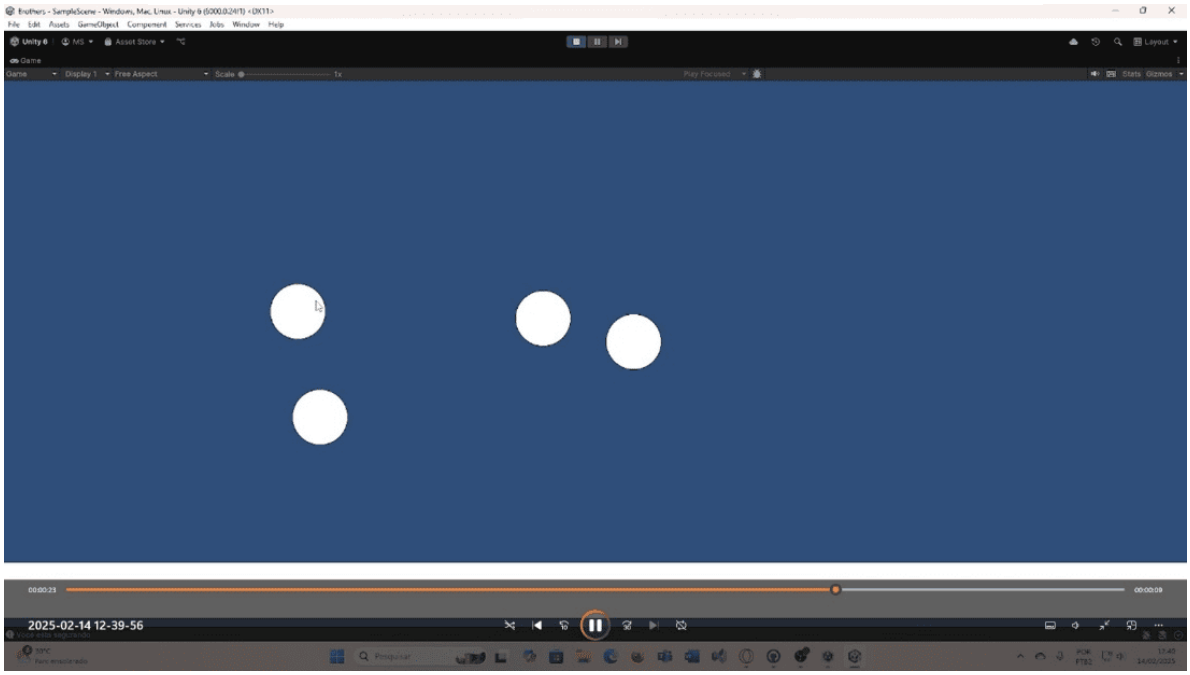
Figura 4 - teste de protótipo de letras



Fonte: do autor.

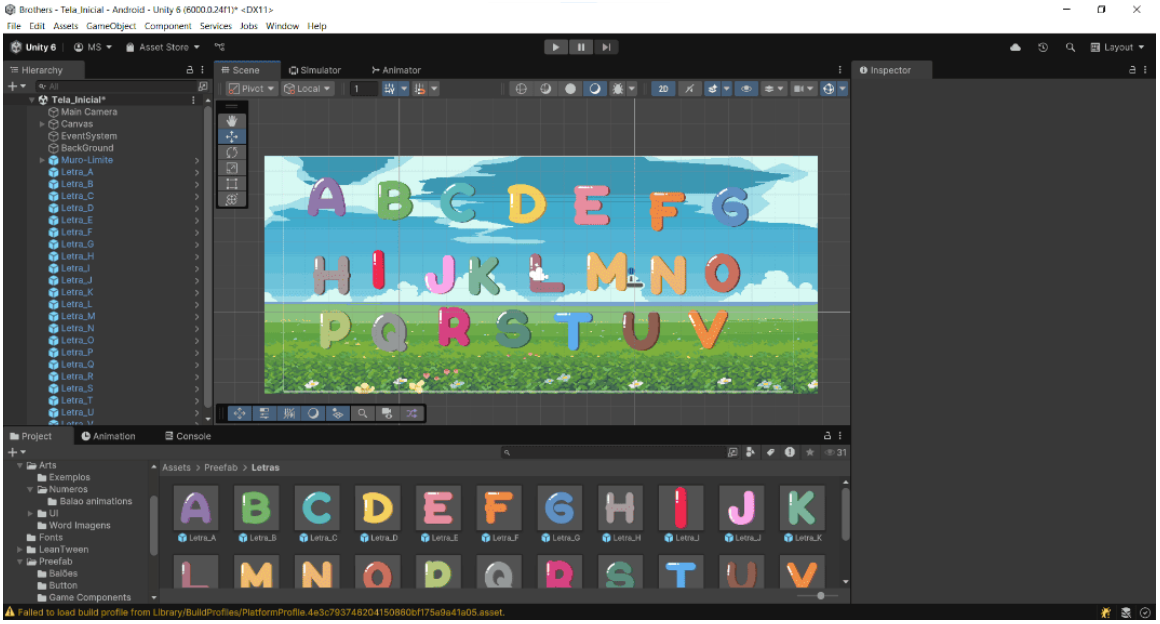
Com o código das letras completo, para diferenciar cada letra, foi necessário colocar a arte de cada no Sprite Renderer dos objetos, nomear cada letra no código e salvar seu *prefab* na pasta do projeto para que a mesma letra pudesse ser usada várias vezes, evitando a necessidade de configurá-la novamente, facilitando posteriormente a criação das fases. Para melhor destaque das letras, foi colocado um plano de fundo colorido e tranquilo, evocando a sensação de paz e paciência. Suavizou-se, dessa forma, o estresse do aluno caso erre a formulação da palavra.

Figura 5 - configuração de letras (Imagem e Som)



Fonte: do autor.

Figura 6 - todas as letras configuradas e suas prefabs salvos



Fonte: Do Autor

Após todas as letras finalizadas, foram feitas as caixas de verificação. Cada uma delas aceitará somente a letra especificada, fazendo a comparação de nomes quando a colisão é detectada entre a caixa de verificação e a letra. Sendo assim, deu-se início à programação, primeiro definindo as variáveis da classe Check Letras:

```
public char Letra_Verificada;
 private Word_manager WM;
 private GameObject Letra_errada;
```

Essas três variáveis são responsáveis pelo funcionamento da caixa de verificação. A variável do tipo char, denominada Letra_Verificada, determina qual letra será aceita pela caixa de verificação. A variável Word_Manager é responsável por enviar ao componente de controle de palavras a informação de que a letra foi posicionada corretamente. Esse componente possui um vetor de valores booleanos que registra o acerto de cada letra da palavra. Por fim, a variável Letra_Errada, do tipo GameObject, armazena o objeto correspondente à letra quando o aluno a posiciona incorretamente. Dessa forma, ao verificar se a palavra foi montada corretamente, o sistema identifica as letras em posições incorretas e aplica um impulso para ejetá-las da área de montagem.

Após a definição dessas variáveis, iniciou-se a implementação das funções utilizadas pela caixa de verificação, sendo duas delas disponibilizadas pela API de colisão da Unity.

```
private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.GetComponent<Letras>().Nome_Letra == Letra_Verificada)
        {
            WM.Acerto();
            col.gameObject.GetComponent<Letras>().SomLetra();
        }
        else
        {
            Letra_errada = col.gameObject;
        }
    }
```


A função “OnTriggerEnter2D” é própria da Unity, a qual reconhece e detecta qualquer objeto com colisão e física aplicada dentro de uma determinada área. Por meio dessa função, é possível verificar qual script do objeto que adentrou a área e verificar suas variáveis, o que se mostra extremamente útil em nossa aplicação. O código acima pega o script do objeto que adentrou na área e faz a comparação com o nome da letra e a letra verificada. Caso esteja na posição correta, executará a função “Acerto()”, do “Word_manager” que insere um dado True no vetor de booleanas. Caso contrário, o objeto da letra não correspondente será armazenada na variável “Letra_Errada” para futuramente manipular a física desse objeto para ser ejetado.

```
private void OnTriggerExit2D(Collider2D col)
{
    if (col.gameObject.GetComponent<Letras>().Nome_Letra == Letra_Verificada)
    {
        WM.Erro();
    }
    else if (col.gameObject.GetComponent<Letras>().Nome_Letra != Letra_Verificada)
    {
        Letra_errada = null;
    }
}
```

Diferentemente da “OntriggerEnter2D”, a “OnTriggerExit2D” é uma função que detecta a saída de um determinado objeto de uma área. O código acima faz justamente o inverso do código anterior. Caso a letra correta saia da área determinada, será executada a função “Erro()”

do “Word-manager” que vai inserir o valor False no vetor de booleanas e se, por um acaso, for retirada da área a letra incorreta, será atribuída à variável “Letra_errada” o valor nulo, já que a letra não se encontra mais na área incorreta.


```
public void expulsar_Letra()
{
   if (Letra_errada != null)
   {
      Letra_errada.gameObject.GetComponent<Rigidbody2D>().AddForce(Vector3.up * 9f,
ForceMode2D.Impulse);
   }
}
```

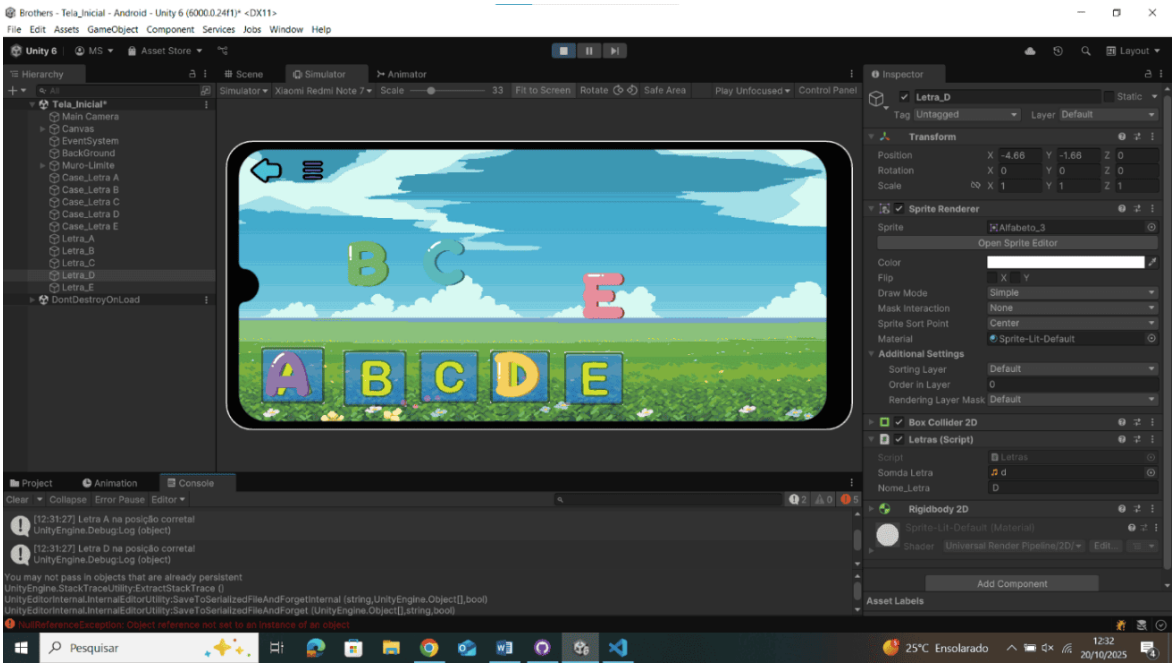
A função ExpulsarLetra é responsável por aplicar um impulso físico ao objeto correspondente à letra posicionada incorretamente, projetando-o para fora da caixa de verificação. Inicialmente, a função verifica se existe um objeto armazenado na variável Letra_Errada (Letra_Errada != null). Em caso afirmativo, o objeto recebe um impulso por meio de seu componente de física, sendo lançado para cima.

Com a implementação do código concluída, foi possível iniciar os testes para verificar seu funcionamento. Para isso, utilizou-se uma mensagem de confirmação no Console da Unity e, com o objetivo de agilizar o processo de desenvolvimento, foi adicionada a arte correspondente a cada caixa de verificação.

Figura 7-prefabs das caixas de verificação

Fonte: do autor.

Figura 8 - teste de colisão e verificação de letras



Fonte: do autor.

Com a parte das letras e as check boxes prontas, iniciou-se o desenvolvimento do word_manager, que ficará responsável de verificar se a palavra foi formada corretamente, de modo que finalize a fase e a opção de o aluno voltar para o menu inicial. Sendo assim, deu-se início ao código definindo as suas variáveis:

```
public bool[] Sequencia;
private RectTransform Vitoria_Panel;
public Sprite[] Fotos;

private GameObject ReturnButton;
private AudioClip Yay, Tente_Denovo;
public Image Imagem_Exemplo;
private int num;
private int indexImage;
private Check_Letras[] Letras_Erradas;
```

Primeiramente, a variável Sequencia é a mais importante do script. Trata-se de um vetor de valores booleanos cujo tamanho é definido pela quantidade de letras da palavra. Dessa forma, sempre que uma letra é posicionada corretamente em uma caixa de verificação, uma função é executada para atribuir o valor true à posição correspondente do vetor Sequencia.

A variável Vitoria_Panel é responsável por referenciar o painel de vitória exibido quando o jogador completa corretamente a sequência da palavra e aciona o botão de verificação. Nesse momento, uma mensagem de parabéns é apresentada ao usuário. Em seguida, a variável Fotos, um vetor do tipo Sprite, armazena as imagens que servirão como referência visual para a criança durante a montagem da palavra. Já a variável ReturnButton, do tipo GameObject, é utilizada para desativar o botão de retorno após a conclusão correta da palavra.

As variáveis Yay e Tente_Denovo, ambas do tipo AudioClip, armazenam os áudios utilizados como retorno ao jogador. O áudio Yay é reproduzido quando a criança monta corretamente a palavra, enquanto o áudio Tente_Denovo é reproduzido quando a sequência está incompleta ou incorreta.

A variável imagem_exemplo, do tipo Image, é responsável por exibir os sprites na tela do dispositivo móvel. As variáveis num e IndexImage, do tipo inteiro, são utilizadas como índices para determinar qual imagem será exibida e a quantidade de letras que compõem a palavra. Por fim, a variável Letras_Erradas é responsável por percorrer todas as caixas de verificação e acionar a função que expulsa as letras posicionadas incorretamente, aplicando um impulso físico sobre elas.

Após a definição dessas variáveis, iniciou-se a implementação das funções responsáveis pelo funcionamento do sistema.

```
void Start()
    {
        num = PlayerPrefs.GetInt("QTD_Letras");
        indexImage = PlayerPrefs.GetInt("IDX_Imagem");

        AC = Object.FindFirstObjectByType<AudioControl>();
        ReturnButton = Object.FindFirstObjectByType<Return_Script>().gameObject;
        Vitoria_Panel = GameObject.Find("Parabens").GetComponent<RectTransform>();

        Letras_Erradas = FindObjectsOfType<Check_Letras>();

        Sequencia = new bool[num];
        Imagem_Exemplo.sprite = Fotos[indexImage];

        for (int i = 0; i < Sequencia.Length; i++)
        {
            Sequencia[i] = false;
        }
```

```
}
```

A função “Start” é oferecida pela Unity e executa comandos quando o primeiro frame é renderizado na tela. Dentro dessa função, foram definidas e instanciadas as variáveis, preenchendo o vetor de booleanas com false.


```
public void Acerto()
{
   for (int i = 0; i < Sequencia.Length; i++)
   {
      if (Sequencia[i] == false)
      {
         Sequencia[i] = true;
         break;
      }
   }
}
```

A função “Acerto” passa pelo vetor sequencial e, no primeiro false que ele encontra, substitui-o por true, encerrando o loop com o break.

```
public void Erro()
{
   for (int i = 0; i < Sequencia.Length; i++)
   {
      if (Sequencia[i] == true)
      {
         Sequencia[i] = false;
         break;
      }
   }
}
```

A função “Erro” faz o contrário da função “Acerto” ela marca com false a primeira posição com true que ela encontrar e quebra o loop com break.

```
private bool Checagem()
{
```

```
        for (int i = 0; i < Sequencia.Length; i++)
        {
            if (Sequencia[i] == false)
                return false;
        }
        return true;
    }
```

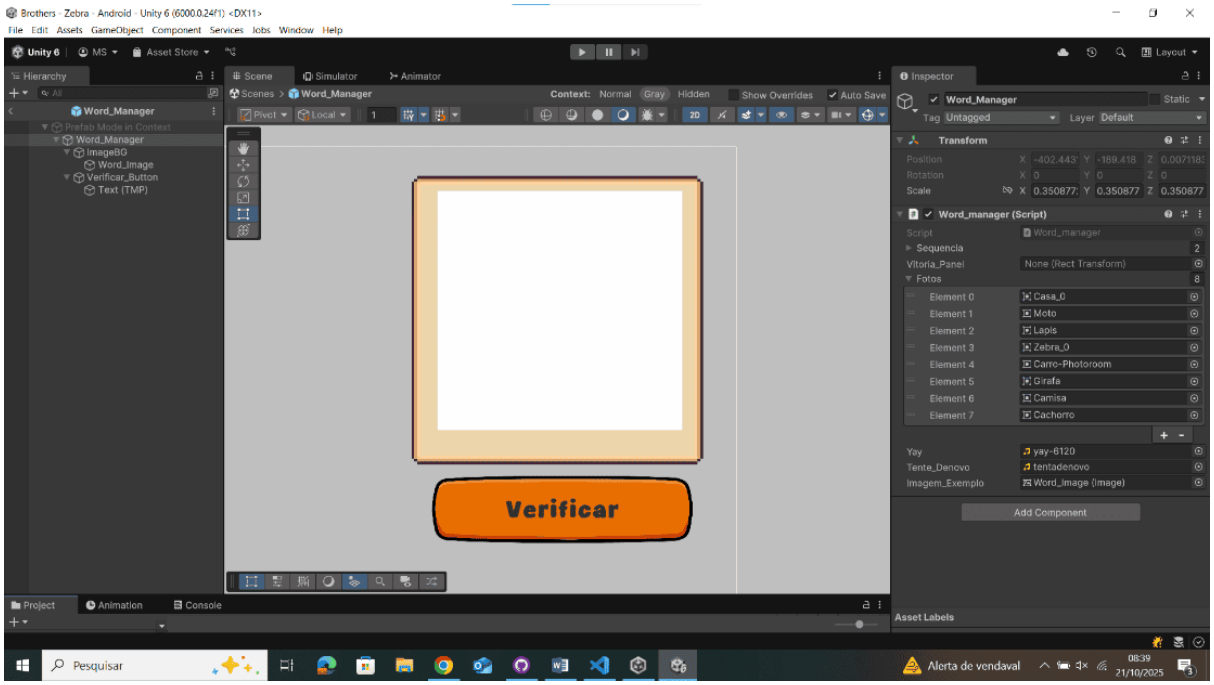
A função Checagem(), que retorna um valor booleano, é responsável por percorrer todo o vetor Sequencia e verificar se todas as posições possuem o valor true. Caso encontre algum elemento com o valor false, a função interrompe a verificação e retorna false. Caso contrário, após percorrer todo o vetor, retorna true, indicando que a sequência foi completada corretamente.

```
public void Verificacao()
{
    bool Correto = Checagem();
if (Correto)
    {
        Vitoria_Panel.LeanMoveY(50f, 0.5f);
        ReturnButton.SetActive(false);
        AC.Tocar_SFX(Yay);
    }
    else
    {
        AC.Tocar_SFX(Tente_Denovo);
        for (int i = 0; i < Letras_Erradas.Length; i++)
        {
            Letras_Erradas[i].expulsar_Letra();
        }
    }
}
```

Por fim, implementou-se a função de verificação, responsável por realizar a validação final da sequência montada pelo jogador. Essa função verifica o resultado obtido e reproduz o áudio correspondente ao acerto ou ao erro, conforme o desempenho da criança. Além disso, ela percorre as caixas de verificação para identificar as letras posicionadas incorretamente e aciona a função responsável por expulsá-las mediante a aplicação de um impulso físico.

Com a implementação de todas as funcionalidades concluída, o script pôde ser atribuído ao objeto correspondente, permitindo a realização dos testes de funcionamento e o desenvolvimento das interfaces do jogo.

Figura 9 - aplicação do script e a arte da interface

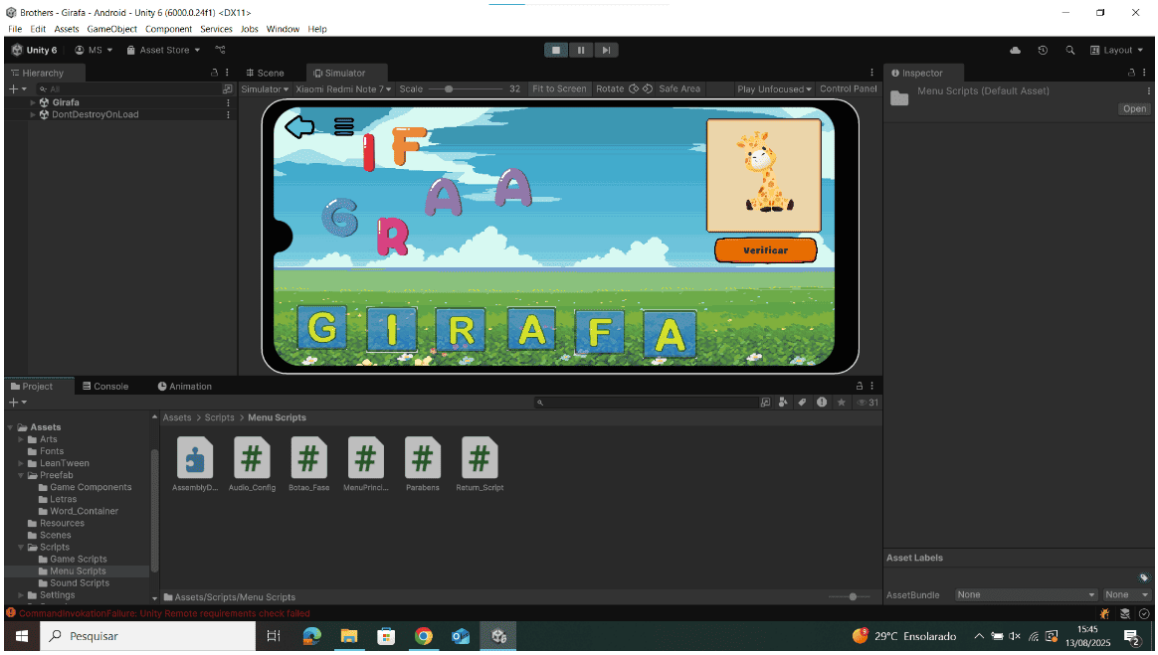


Fonte: do autor.

1.3 Criando as fases e menus

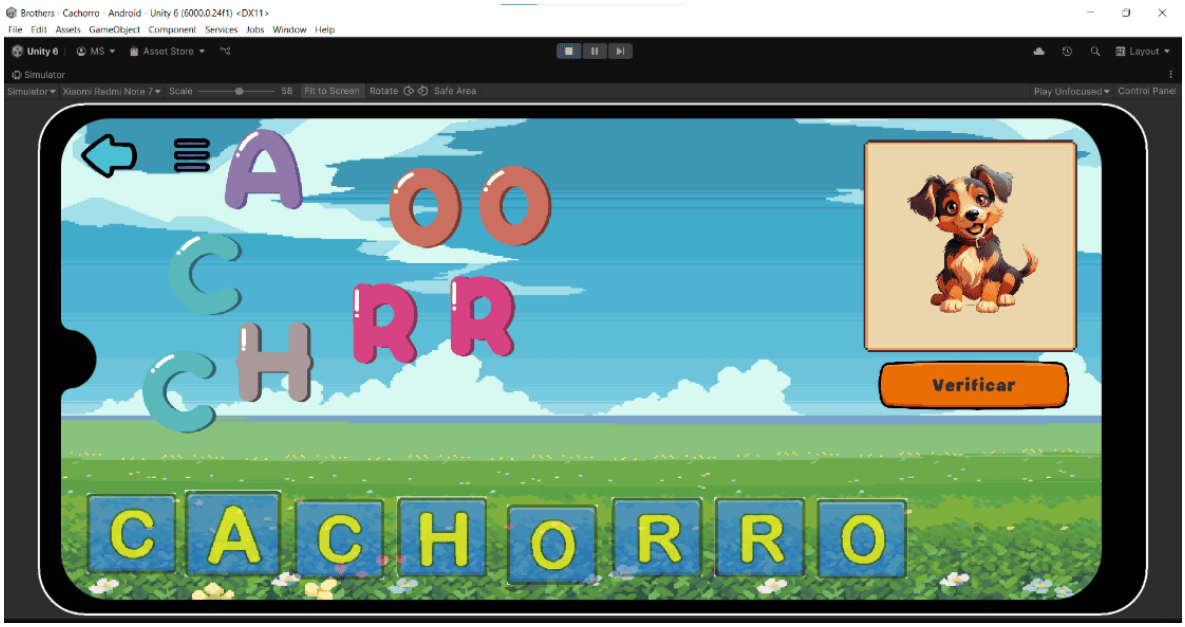
Com a implementação do sistema-base concluída, iniciou-se a etapa de montagem das fases e da seleção das palavras, organizadas em uma progressão gradual de dificuldade. As atividades tiveram início com palavras de quatro letras e evoluíram para palavras com maior quantidade de letras e diferentes regras gramaticais. Ao todo, foram desenvolvidas oito fases.

Figura 10 - fase do Cachorro



Fonte: do autor.

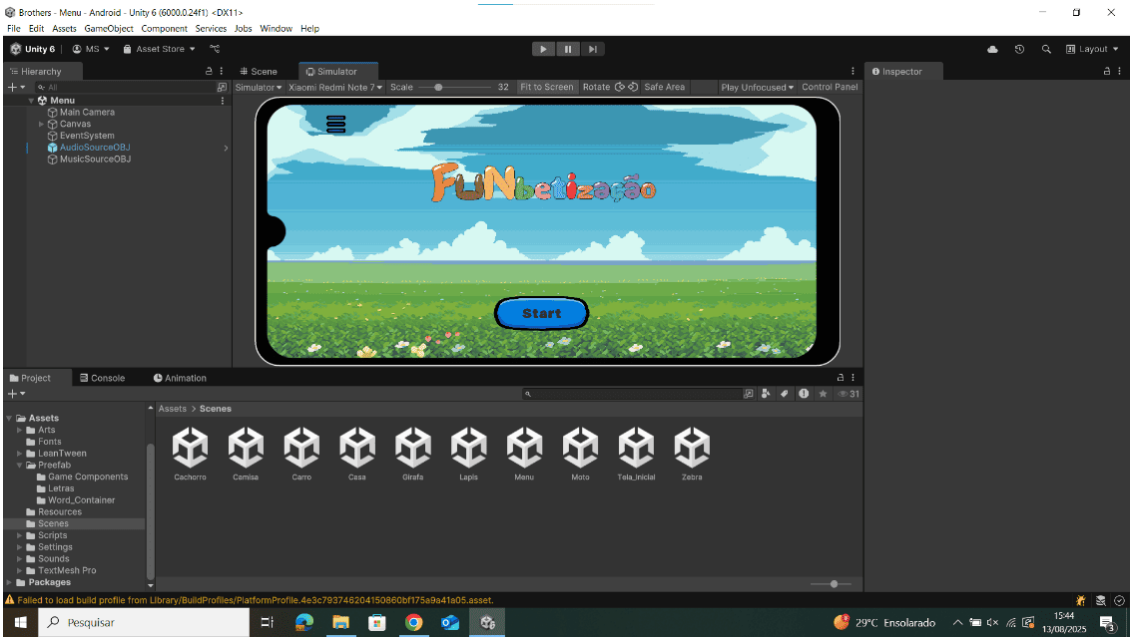
Figura 11 - montagem da fase da girafa



Fonte: do autor.

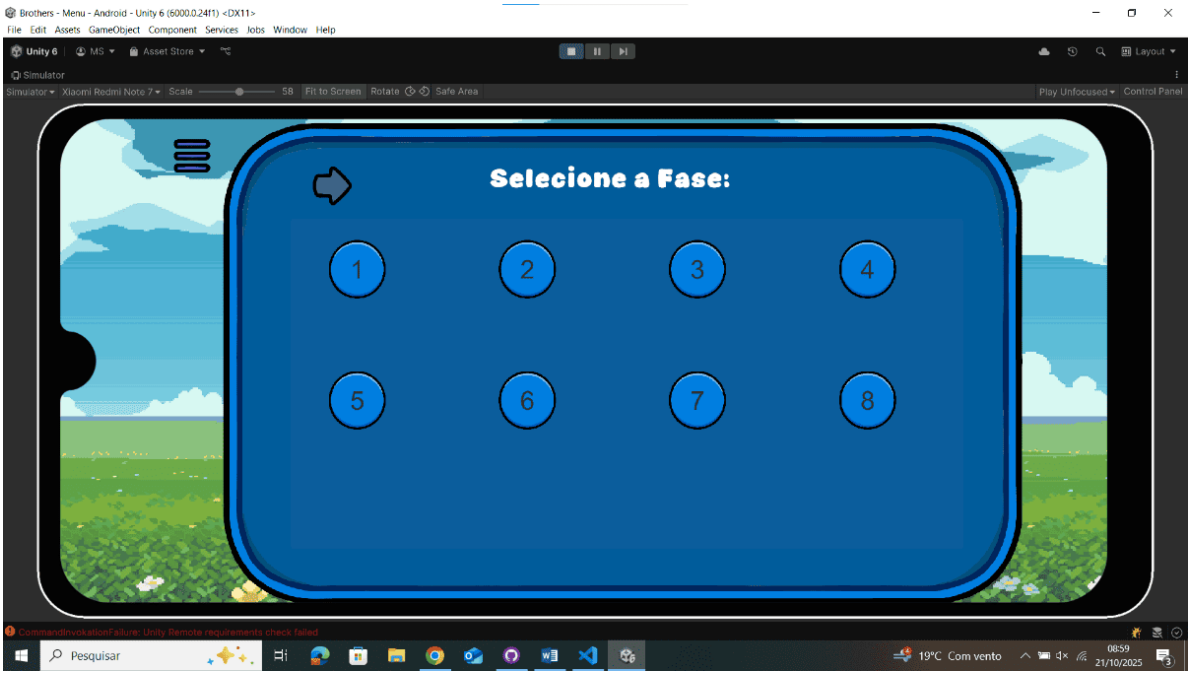
Tendo finalizado as oito fases, foi necessário desenvolver um menu intuitivo e simples, para que o cuidador ou a criança fosse capaz de entender rapidamente, evitando poluição visual ou muita informação desnecessária na tela.

Figura 12 - tela principal



Fonte: do autor.

Figura 13 - tela de seleção de fases



Fonte: do autor.

2 RESULTADOS

Após a conclusão da versão inicial do jogo, foi possível portá-lo para um dispositivo móvel e realizar testes com um aluno do Colégio Unifev. Os testes ocorreram em um ambiente

controlado, com o acompanhamento e o auxílio da cuidadora da criança durante toda a atividade.

No primeiro contato com o jogo, o aluno demonstrou certo estranhamento em relação à mecânica de interação, pois estava habituado a utilizar o celular principalmente para assistir a vídeos curtos, realizando apenas movimentos de deslizamento na tela. Com a orientação da cuidadora, a criança compreendeu que era necessário tocar na tela para selecionar e iniciar uma fase.

Durante a execução das atividades, observaram-se dificuldades relacionadas à coordenação motora, uma vez que o aluno apresentava dificuldade para selecionar e arrastar corretamente as letras, necessitando, inicialmente, do auxílio da cuidadora para identificar os locais de toque na tela. No entanto, à medida que interagia com o jogo, a criança passou a montar as palavras de forma mais autônoma, cometendo menos erros e demandando cada vez menos suporte. Para um primeiro contato com o produto, destacou-se a rapidez com que o aluno compreendeu seu funcionamento e passou a associar os sons das letras às suas respectivas representações visuais.

Durante os testes, também foram identificados alguns aspectos passíveis de melhoria. Observou-se que a criança demonstrava frustração quando errava a sequência das letras ou quando encontrava dificuldades para arrastá-las, em razão do reduzido tamanho da área de colisão dos objetos. Dessa forma, verificou-se a necessidade de ampliar essa área de interação, tornando a manipulação das letras mais fácil e intuitiva.

Além disso, constatou-se que o jogo contribuiu para o reconhecimento de palavras e objetos por meio das imagens apresentadas na tela, bem como proporcionou à criança uma forma diferente de interação com o dispositivo móvel, além do consumo passivo de vídeos. Os testes também permitiram identificar pequenos problemas de implementação e apontaram possibilidades de expansão do conteúdo pedagógico do jogo, com base nas sugestões apresentadas pela cuidadora.

3 DISCUSSÃO

Os resultados observados com o uso do Funbetização corroboram a literatura existente, a qual sugere a gamificação como uma ferramenta valiosa e eficaz no ensino de crianças com Transtorno do Espectro Autista (TEA). A estrutura do jogo, baseada em pistas visuais, repetição controlada e reforços positivos imediatos, demonstrou favorecer a atenção e o engajamento dos participantes. Tais achados estão alinhados ao que aponta Baracho (2024), ao destacar que a

previsibilidade e o suporte visual são fundamentais para o aprendizado e o conforto cognitivo de indivíduos com TEA.

Além disso, o foco em elementos visuais aproveita a alta capacidade de processamento sensorial desse público, característica central nas discussões de Miliausks e Pinheiro (2024) sobre intervenções educacionais bem-sucedidas. Por outro lado, o desenvolvimento de um software educativo exige rigor técnico. Conforme os princípios de Sommerville (2018), a engenharia de software deve prever requisitos específicos do usuário final para garantir a robustez do sistema.

No entanto, desafios práticos foram identificados, como a frustração do aluno em momentos de erro ou dificuldades de coordenação motora relacionadas ao tamanho das áreas de colisão. Essas observações alinham-se com as advertências de Cassol (2022) sobre a importância de testes de usabilidade rigorosos e a necessidade de considerar a sensibilidade sensorial e as habilidades motoras finas no *design* de jogos voltados para a diversidade funcional. Portanto, o projeto valida a abordagem tecnológica, ao mesmo tempo que utiliza as bases da lógica de programação e ferramentas digitais — como as exploradas por Lima (2020) — para direcionar ajustes futuros que aprimorem a acessibilidade e a experiência do usuário.

CONCLUSÃO

Com esta pesquisa, conclui-se que a gamificação constitui uma ferramenta de grande potencial para a educação e a inclusão digital, contribuindo não apenas para o processo de aprendizagem, mas também para o desenvolvimento da coordenação motora. Os testes realizados demonstraram que a criança se adaptou rapidamente às mecânicas do jogo e foi capaz de distinguir as letras tanto pelos sons quanto por suas representações visuais.

Considerando as particularidades do processo de aprendizagem de crianças com Transtorno do Espectro Autista (TEA), observou-se que a repetição das atividades desempenha um papel importante na consolidação do reconhecimento das letras e na associação entre seus sons e suas formas. Nesse contexto, o uso de recursos tecnológicos adequados pode favorecer significativamente o desenvolvimento dessas habilidades.

Embora o processo de ensino de crianças com TEA apresente desafios específicos, verificou-se que esses podem ser minimizados com a utilização de estratégias pedagógicas apropriadas, tecnologias educacionais e o acompanhamento de profissionais capacitados. Para isso, é fundamental que os educadores possuam conhecimentos sobre as características do

transtorno e compreendam que sua manifestação varia de acordo com as necessidades individuais de cada criança.

Ressalta-se que o jogo desenvolvido não substitui a atuação do educador, mas configura-se como uma ferramenta de apoio ao processo de ensino e aprendizagem. A presença de um profissional continua sendo indispensável para orientar a criança durante as atividades, fornecer suporte quando necessário e potencializar o aproveitamento pedagógico. Os estímulos visuais e sonoros presentes no jogo mostraram-se relevantes para manter a atenção do aluno e favorecer seu engajamento, evidenciando o potencial da tecnologia como recurso complementar na educação inclusiva.

REFERÊNCIAS

AMERICAN PSYCHIATRIC ASSOCIATION**. Manual diagnóstico e estatístico de transtornos mentais**: DSM-5. 5. ed. Porto Alegre: Artmed, 2014.

BARACHO, Marcos. 7 aspectos fundamentais do Transtorno do Espectro Autista (TEA) que todos deveriam saber. **Autismo VR**, 1 nov. 2024. Disponível em: https://autismovr.com.br/oque-e-o-autismo/. Acesso em: 29 nov. 2024.

CASSOL, Vinícius. **Programação aplicada a games**. Curitiba: InterSaberes, 2022.

LIMA, Victor. **Curso de C#**: aprenda o essencial em 5 horas. YouTube, 18 set. 2020. Disponível em: https://www.youtube.com/watch?v=PKMm-cHe56g. Acesso em: 21 out. 2025.

MILIAUSKAS, Claudia; PINHEIRO, Pedro. Autismo (Transtorno do Espectro Autista). **MD.Saúde**, 2024. Disponível em: https://www.mdsaude.com/psiquiatria/autismo/. Acesso em: 29 nov. 2024.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson Education do Brasil, 2018.