CONSEQUÊNCIAS – UM JOGO DE EXPLORAÇÃO UTILIZANDO REALIDADE VIRTUAL SOBRE CONSCIENTIZAÇÃO AMBIENTAL

Vitor Hugo Helmbrecht, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

vhelmbrecht@furb.br, dalton@furb.br

Resumo: *Este artigo apresenta o processo de desenvolvimento e avaliação de um jogo para plataformas de realidade virtual e com foco na conscientização ambiental do jogador. O jogo foi desenvolvido utilizando o motor gráfico Unity3D, além da linguagem C# e as ferramentas XR Interaction Toolkit, Cinemachine, Timeline e NavMesh. Tanto para o desenvolvimento quanto para a realização dos testes foi utilizado o dispositivo Meta Quest 2. Os resultados obtidos foram satisfatórios, de forma que as histórias do jogo se mostraram de fácil compreensão para os jogadores e a jogabilidade foi agradável, apesar de difícil. Por fim, concluiu-se que o jogo cumpriu com os objetivos aos quais foi proposto, propiciando uma forma de ensinar conscientização ambiental através de um jogo de realidade virtual.*

Palavras-chave*: Realidade virtual. Jogo. Exploração. Unity. Conscientização ambiental.*

1 INTRODUÇÃO

O aquecimento global é um aumento da temperatura média superficial global (SILVA; PAULA, 2009). Esse fenômeno pode acontecer por algumas razões diferentes, sejam elas internas ou externas. De acordo com Silva e Paula (2009), fatores internos do aquecimento global são complexos e inconstantes, além de dependerem de diversas variáveis diferentes, como a atividade solar e o vulcanismo. Já os fatores externos são causados por ações como a queima de combustíveis fósseis ou queimadas. Os autores apresentam fatores tanto externos quanto internos que vêm realizando essas alterações climáticas ao longo de séculos, mas uma coisa é afirmada: o ser humano possui um impacto sobre a mudança climática do planeta.

Conscientização ambiental está relacionado às ações que o indivíduo exerce que estão vinculadas ao meio ambiente (SÁNCHEZ; LAFUENTE, 2010). Ao mesmo tempo, educação ambiental já é no Brasil um dos temas contemporâneos transversais da Base Nacional Comum Curricular (BNCC), ou seja, ele deve ser abordado durante o ensino, mas sem ser tratado como uma imposição de conteúdo (BRASIL, 2022). Isso demonstra como a conscientização ambiental já está sendo abordada no Brasil, além de ser um assunto importante para que possamos diminuir o impacto do ser humano sobre as mudanças negativas que ocorrem no planeta.

O mercado de videogames mundial gerou 196.8 bilhões de dólares em 2022 (NEWZOO, 2023), enquanto o mercado de cinema gerou 77 bilhões no mesmo período (IBISWORLD, 2022), o que demonstra como as pessoas estão cada vez mais interessadas em videogames. Junte a isso o fato de que o uso de equipamentos de realidade virtual teve um grande aumento recentemente (TAO *et al*., 2021), e é possível perceber a fundação de um mercado promissor.

Assim como Tao *et al*. (2021) falaram, não é apenas para uso pessoal que as pessoas começaram a adotar mais os videogames. Existe também o uso voltado para, por exemplo, o tratamento de condições clínicas. Visto que esse mercado continua crescendo e que a realidade virtual possui múltiplas funcionalidades, esse trabalho tem como objetivo construir um jogo para plataformas de realidade virtual com o foco na conscientização ambiental do jogador. O objetivo específico do jogo é: incorporar o tema de conscientização ambiental.

2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção são apresentados os assuntos que fundamentam o desenvolvimento do jogo. A primeira subseção trata sobre realidade virtual, a segunda subseção trata sobre a conscientização ambiental e a terceira subseção trata sobre os trabalhos correlatos.

2.1 REALIDADE VIRTUAL

Segundo Azuma (1997, p.2), Realidade Virtual (RV) faz com que, enquanto imerso, o usuário não possa ver o mundo que está ao seu redor. Essa tecnologia coloca o usuário em um mundo à parte, completamente novo. De acordo com Ding, Li e Cheng (2020), a RV é uma tecnologia que combina uma variedade de outras tecnologias, como computação, multimídia, processamento de imagem, entre outros, e por fim cria uma tecnologia que, por sua vez, está

dentro do campo de computação e simulação. Os autores ainda afirmam que a tecnologia de RV inclui a simulação de ambientes, percepções, habilidades naturais (como a rotação da cabeça, olhos, gestos etc.) e equipamentos sensoriais.

Tao *et al*. (2021) comenta que recentemente ocorreu um aumento no uso de RV não apenas para uso pessoal, como também no tratamento de diferentes condições clínicas. Os autores afirmam que isso se deve ao fato de os *head-mounted displays* (HMD) de alta qualidade, que são os equipamentos através dos quais é possível ter uma experiência RV mais imersiva, estão se tornando cada vez mais acessíveis. A Figura 1 exemplifica essa normalização de HMDs de alta qualidade. A propaganda da empresa Meta é realizada com a utilização de pessoas aproveitando a experiência de RV com HMDs em suas casas.

Figura 1 – Propagando do Meta Quest 2

Fonte: Meta (2023).

2.2 CONSCIENTIZAÇÃO AMBIENTAL

Sánches e Lafuente (2010) definem conscientização ambiental como o conceito que refere a fatores especificamente psicológicos que estão relacionados à propensão do indivíduo de realizar comportamentos que favorecem o meio ambiente. Homburg e Stolberg (2006) citam quatro exemplos de comportamentos que podem ser classificados como favorecedores do meio ambiente, sendo eles: ativismo ambiental; comportamentos não-ativistas em esferas públicas; ambientalismo do setor privado; e comportamentos em organizações.

Alguns exemplos mais concretos de ações que favorecem o meio ambiente, de acordo com os exemplos que Homburg e Stolberg (2006) oferecem, são: envolvimento de organizações ambientais; petições; poupar energia; comprar objetos feitos com materiais reciclados; design de produtos.

Por exemplo, o design de produtos, pode se tornar uma ação que favorece o meio ambiente, pois o seu objetivo é diminuir o impacto que os produtos desenvolvidos geram no meio ambiente. Nesse caso, alguns produtos estão aos poucos tendo seus designs melhorados. Nascimento, Santos e Silva (2022) explicam que o bioplástico é uma alternativa para o plástico utilizado atualmente. Esse material, de acordo com os autores, pode se desintegrar em questão de 180 dias, desde que estejam em condições adequadas. Dessa maneira, o bioplástico ajuda a aumentar a vida útil de aterros sanitários, além de poder virar adubo orgânico após desintegrado.

2.3 TRABALHOS CORRELATOS

São apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro está detalhado no Quadro 1 e o trabalho se trata do jogo CidadaniaAR (NIENOW, 2019), que foi desenvolvido utilizando realidade aumentada e ilusão de ótica, além de abordar o tema reciclagem. O segundo trabalho, que está sendo apresentado no Quadro 2, explica um pouco mais sobre o jogo Vignettes (SKELETON BUSINESS, 2017), que tem o foco em exploração e na sua utilização de ilusão de ótica. Já o Quadro 3 traz informações sobre o jogo Monument Valley (USTWO GAMES, 2014), que é um jogo para dispositivos móveis que também se utiliza de ilusão de ótica como mecânica principal do jogo para avançar na história.

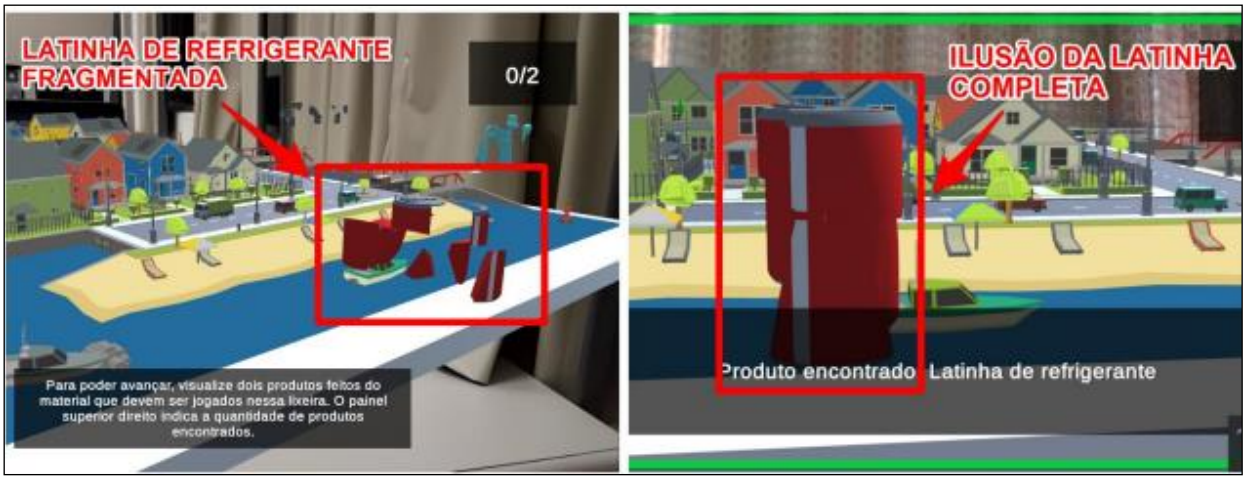
Quadro 1 – CidadaniaAR

Referência	Nienow (2019).
Objetivos	Criar um jogo que faça uso da ilusão de ótica como mecânica principal de jogabilidade.
Principais funcionalidades	Resolução de desafios utilizando ilusão de ótica.
Ferramentas de desenvolvimento	Unity 3D.
Resultados e conclusões	Após a aplicação de questionários, concluiu-se que o objetivo foi alcançado, apesar de pontos a melhorar terem sido apresentados.

Fonte: elaborado pelo autor.

Nienow (2019) desenvolveu o CidadaniaAR, um jogo de puzzle e realidade aumentada focado em ilusão de ótica, incorporando o tema de reciclagem de forma interativa, conforme demonstrado na Figura 2. O jogo foi desenvolvido utilizando o motor gráfico Unity junto da linguagem de programação C#, além do *plugin* para realidade aumentada Vuforia e o software Blender para o desenvolvimento de customizações do jogo em relação a modelos 3D.

Figura 2 – Puzzle de fragmentação e sua resolução



Fonte: Nienow (2019).

Quadro 2 - Vignettes

Referência	Skeleton Business (2017).
Objetivos	Produto comercial.
Principais funcionalidades	Utilização da ilusão de ótica para realizar a exploração da história.
Ferramentas de desenvolvimento	Unity 3D.
Resultados e conclusões	Vignettes recebeu e foi indicado a premiações desde 2015 até 2018, incluindo o prêmio IndieCade EU 2017.

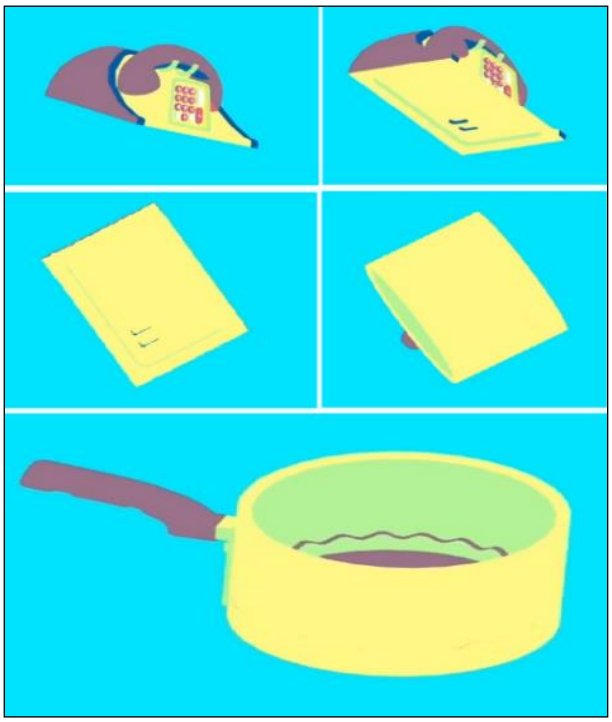
Fonte: elaborado pelo autor.

O estúdio Skeleton Business (2017) desenvolveu Vignettes, um jogo indie comercial, utilizando o motor gráfico Unity. O jogo foi lançado oficialmente em 2017 para iOS e em 2019 para computador, e é focado em exploração com um toque incomum de ilusão de ótica.

Durante o jogo, o jogador deve rotacionar os objetos que aparecem em sua tela até que ele encontre um ângulo que transformará esse objeto em algum outro objeto ou mesmo um animal. Além da rotação, o jogador também pode interagir com quase todos os objetos através de toques/cliques ou alteração do zoom. O que se torna não só um extra interessante para o jogador com interações como o barulho de um rádio ou afagar um gato, como também se torna algo necessário em determinados momentos para liberar objetos escondidos.

A Figura 3 demonstra um exemplo de interação com o objeto dentro do jogo. Como é possível observar, no canto superior esquerda da figura está um telefone em um fundo azul. No canto superior direito da figura, o jogador está rotacionando esse telefone para tentar encontrar o ângulo no qual ele se transforma em algum outro objeto. Conforme demonstrado nas duas imagens que estão no centro da figura, o jogador encontra o ângulo correto no qual ele consegue transformar aquele telefone em outro objeto, que nesse caso é uma panela, que aparece por completa na última imagem da figura.

Figura 3 – Exemplo de transformação de telefone para panela



Fonte: elaborado pelo autor.

Quadro 3 - Monument Valley

Referência	Ustwo Games (2014).
Objetivos	Produto comercial.
Principais funcionalidades	Utilização da ilusão de ótica para resolver enigmas e avançar na história.
Ferramentas de desenvolvimento	Unity 3D.
Resultados e conclusões	Monument Valley recebeu diversas premiações, sendo o principal o prêmio de melhor jogo para dispositivos móveis de 2017 pelo The Game Awards.

Fonte: elaborado pelo autor.

Monument Valley é um jogo indie comercial desenvolvido utilizando o motor gráfico Unity pela empresa USTWO Games (2014). O jogo foi lançado oficialmente em 2014 para iOS e Android, e é focado em puzzles com a utilização principal de ilusões de ótica para formar objetos impossíveis e assim conseguir avançar na história. Alguns trechos do jogo podem ser visualizados na Figura 4.

Figura 4 - Alguns níveis do jogo Monument Valley

Fonte: Monument Valley.

3 DESCRIÇÃO DO JOGO

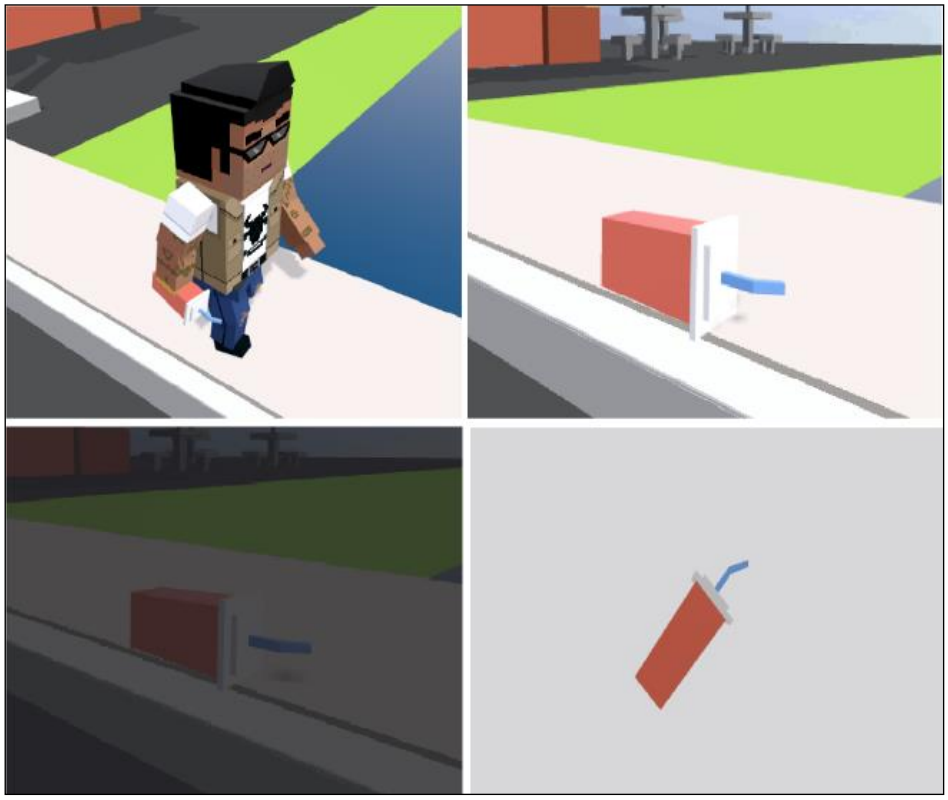
Nessa seção são apresentados os aspectos fundamentais e mais importantes da implementação do jogo e está dividida em duas partes, das quais a primeira explica sobre a visão geral do jogo, enquanto a segunda descreve a implementação do jogo.

3.1 ESPECIFICAÇÃO DO JOGO

O jogo desenvolvido coloca o jogador como observador dos personagens principais e seus mundos. Existem um total de três histórias e dois personagens principais. O jogo também é dividido em dois tipos de interações com o jogador, a primeira sendo onde o jogador apenas acompanha os personagens principais e observa a história se desenrolando através de uma cinemática, e a segunda onde o jogador interage diretamente com determinados objetos.

Cada nível começa com a história do personagem principal se desenrolando, à qual o jogador apenas acompanha enquanto pode observar o cenário ao redor, já que ele está em um ambiente de realidade virtual. Em algumas partes predeterminadas da história é realizada uma transição para o cenário de interação com objetos, conforme demonstrado na Figura 5.

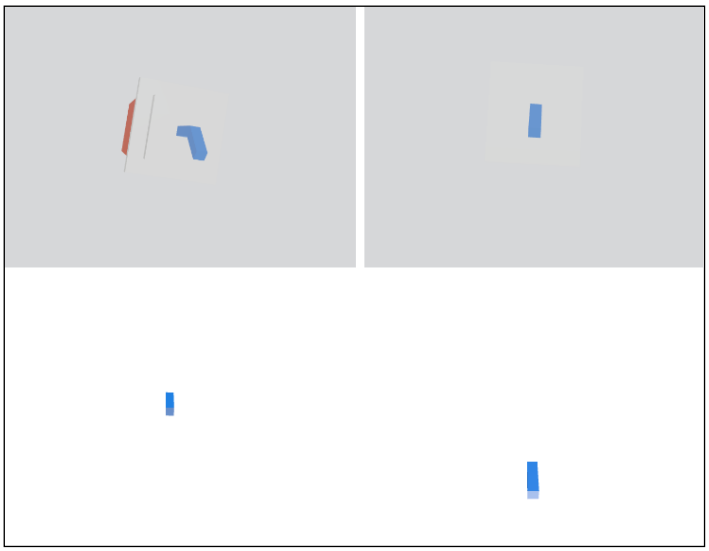
Figura 5 - Transição da cena de história para a cena de interação



Fonte: elaborado pelo autor.

A partir do momento que o jogador estiver na cena de interação com o objeto, ele deverá então utilizar do controle do seu aparelho de realidade virtual para rotacionar o objeto até um ângulo no qual ele consiga formar um novo objeto, assim como demonstrado na Figura 6, onde o jogador deve transformar o canudo do copo de plástico em uma gota de chuva, que iniciará assim a cena 2.

Figura 6 - Transformação do copo em gota d'água e começando a cair



Fonte: elaborado pelo autor.

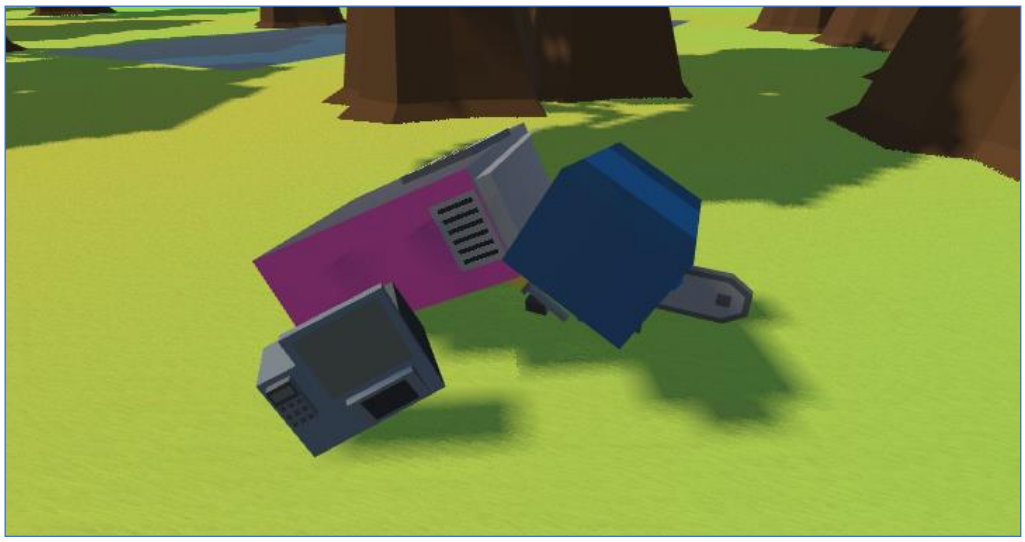
Após a transformação do objeto, a próxima cena de acompanhamento irá iniciar automaticamente, passando assim para a próxima parte da história. O jogo se baseia nessa mecânica para contar suas três histórias de como os atos cometidos pelos personagens principais possuem consequências, sejam elas positivas ou negativas, na vida delas mesmas e de outros seres vivos.

Os roteiros das histórias de cada fase foram desenvolvidos juntamente com o Nando Matheus Rocha, que é um especialista graduado em Ciências Biológicas e mestrando em Ensino de Ciências Naturais e Matemática. A `Fase 1` conta a história de uma pessoa que frequentemente olhava um pássaro que cantava em uma árvore em frente à janela de seu quarto, como ilustrado na Figura 7. Um dia, o monitor do computador dessa pessoa parou de funcionar, e ela foi descartar esse monitor. Acontece que essa pessoa descartou o monitor em uma mata próxima da casa dele. Esse local onde fez esse descarte acabou com o tempo acumulando o lixo eletrônico não só dele, mas também de outras pessoas que fizeram esse descarte inadequado, assim como demonstrado na Figura 8. Perto do local de descarte havia um lago, no qual animais costumavam ir para tomar água. Devido ao descarte inadequado, a água desse lago acabou se contaminando, e o pássaro que cantava na janela do personagem acabou morrendo por tomar água desse lago. Por fim, a história acaba com o personagem olhando pela janela de seu quarto, mas dessa vez, sem nenhum pássaro para cantar.

Figura 7 - Personagem observando o pássaro cantar através da janela do seu quarto

Fonte: elaborado pelo autor.

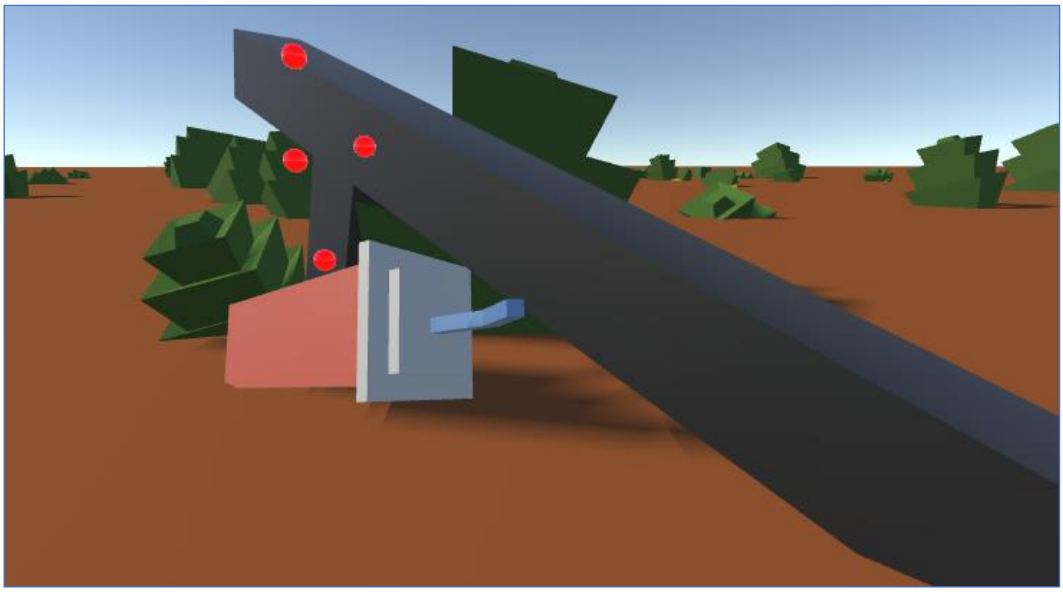
Figura 8 - Pilha de lixo eletrônico em uma área de mata



Fonte: elaborado pelo autor.

A `Fase 2` conta a história de um personagem que estava tomando um refrigerante na rua após ter saído de uma hamburgueria e, ao terminar de tomá-lo, descartou o copo plástico vazio no chão, assim como a Figura 5 aponta. Após uma chuva ocorrer no local, o copo foi levado para o rio que estava próximo e, através das correntezas, acabou chegando no mar. Enquanto o copo está passando pelo mar, é possível ver outros materiais que foram descartados de forma inadequada, até que o copo vai para baixo da água e acaba ficando preso no chão do mar. Aos poucos, o copo acaba soltando microplásticos no mar, conforme demonstrado na Figura 9 (representado pelas esferas vermelhas). Esses microplásticos então acabam sendo levados pelo mar e se fixando em algas, as quais são consumidas por peixes. Esses peixes então são consumidos por peixes maiores, que já possuem múltiplos microplásticos em seu corpo por consumir vários peixes menores infectados. Esses peixes então são pescados por um barco pesqueiro e vendidos em um mercado (Figura 10). O personagem que fez o descarte inicial do copo de plástico acaba então comprando e comendo um desses peixes, e devido a isso, acaba passando mal.

Figura 9 - Microplásticos se soltando do copo no fundo do mar



Fonte: elaborado pelo autor.

Figura 10 - Personagem comprando peixe infectado por microplásticos

Fonte: elaborado pelo autor.

Por fim, a `Fase 3` reapresenta as duas primeiras histórias, mas dessa vez com os descartes dos objetos sendo feitos da forma correta. Primeiramente a segunda história é recontada, e dessa vez o personagem principal descarta o copo vazio em um lixeiro de lixo reciclável. Dessa maneira, o lixo dele foi reciclado em novos materiais de plástico, assim como demonstrado na Figura 11. Já em relação à reapresentação da primeira história, o personagem principal agora descarta o seu monitor em um caminhão de lixo eletrônico, que está representado na Figura 12. Com isso, o caminhão leva o lixo eletrônico para o local correto de descarte e gera dinheiro com a venda dele.

Figura 11 - Itens reciclados

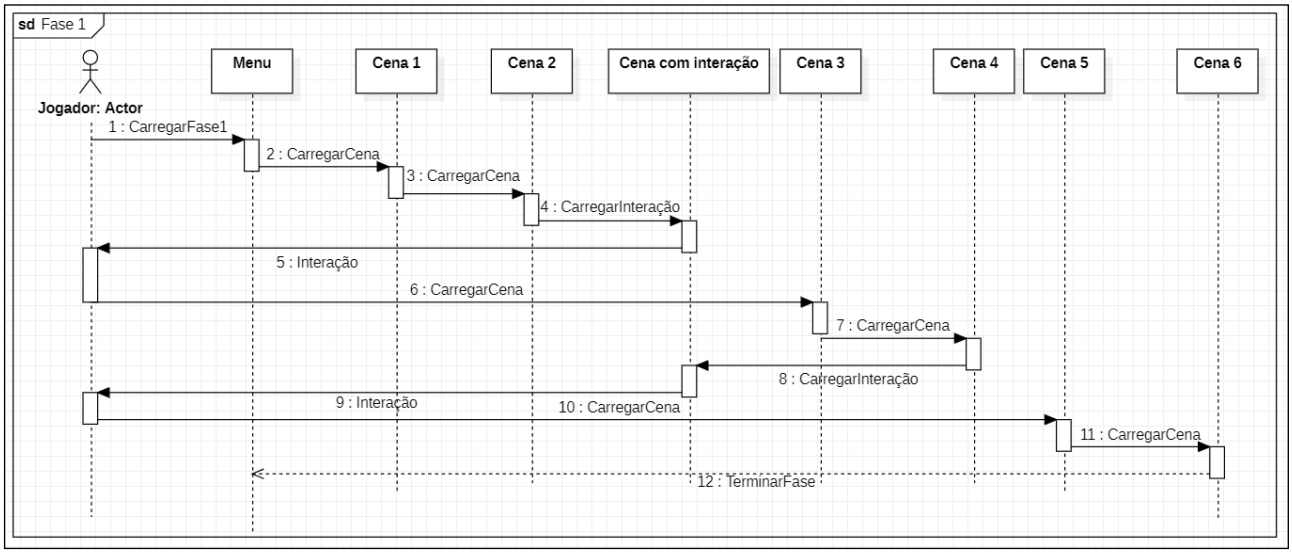
Fonte: elaborado pelo autor.

Figura 12 - Caminhão de lixo eletrônico

Fonte: elaborado pelo autor.

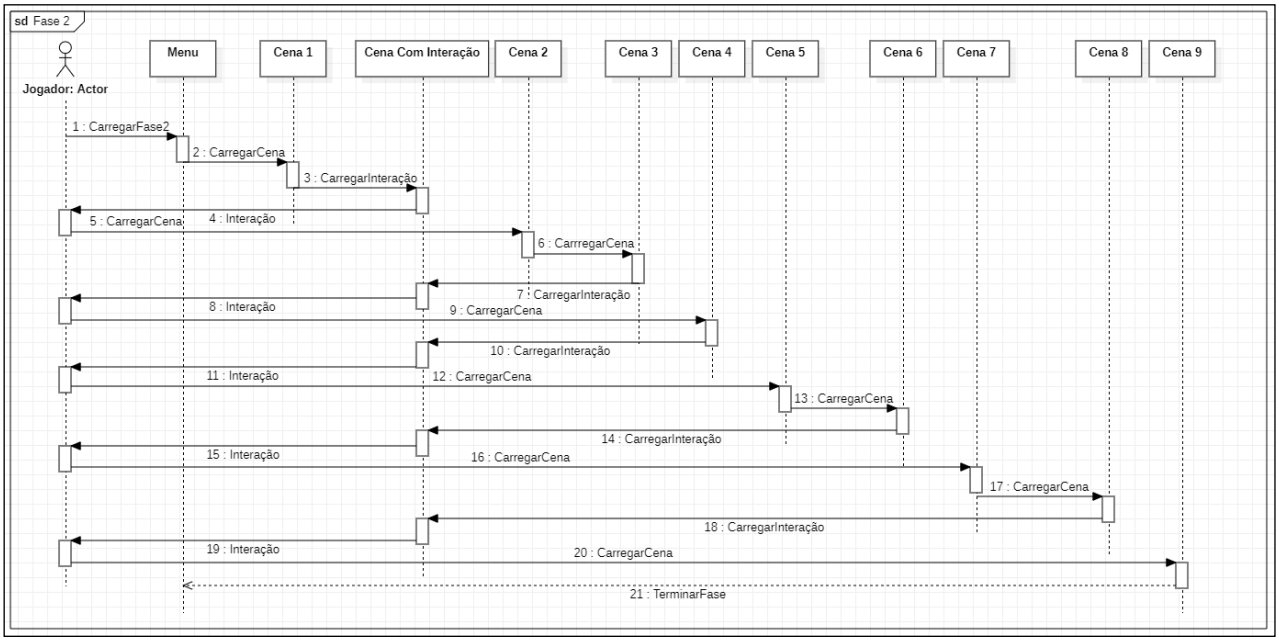
Para elucidar melhor as sequências de cenas que são seguidas em cada fase do jogo, foram criados diagramas de sequências para cada uma, conforme pode ser observado na Figura 13, Figura 14 e Figura 15.

Figura 13 - Diagrama de sequências da `Fase 1`



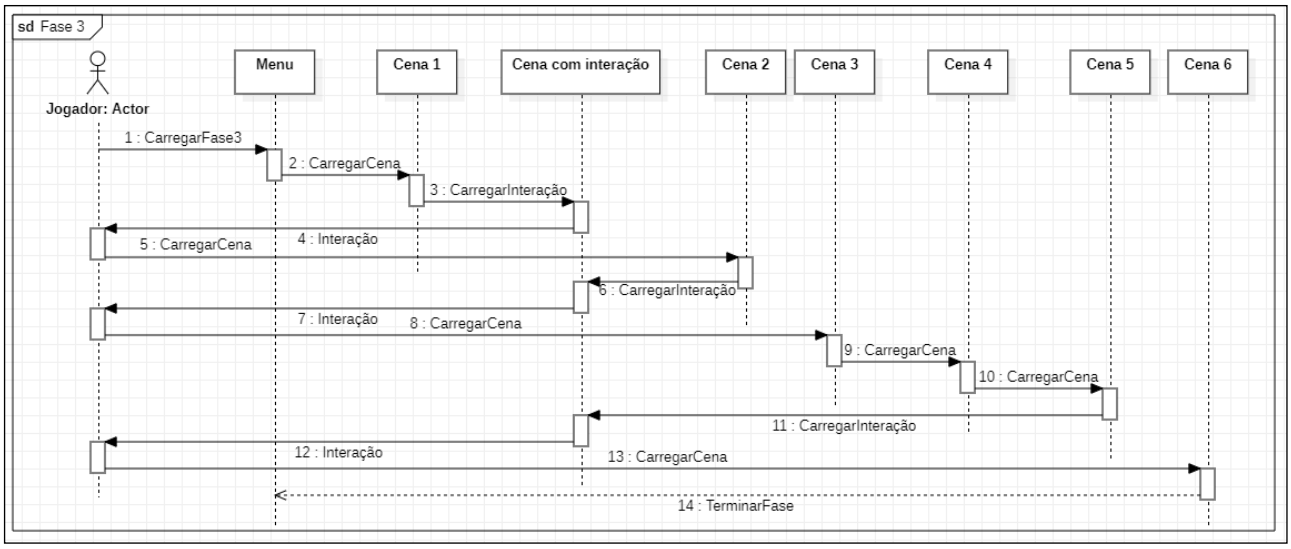
Fonte: elaborado pelo autor.

Figura 14 - Diagrama de sequências da `Fase 2`



Fonte: elaborado pelo autor.

Figura 15 - Diagrama de sequências da `Fase 3`



Fonte: elaborado pelo autor.

3.2 IMPLEMENTAÇÃO

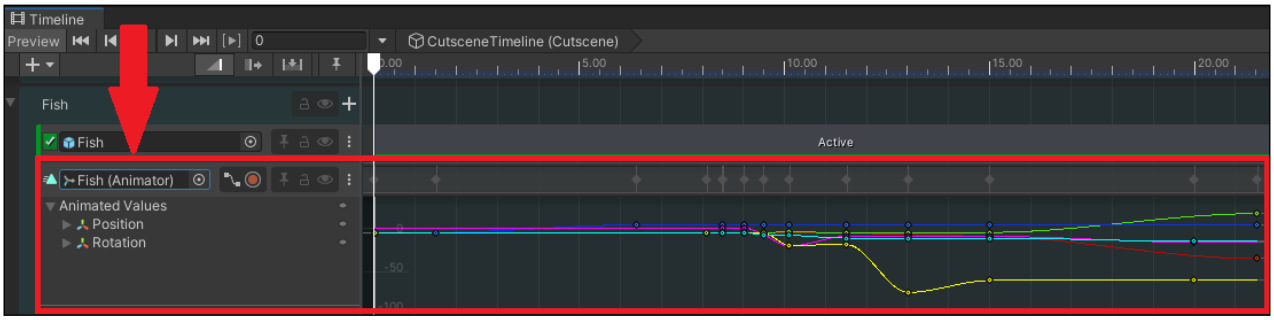
O jogo foi desenvolvido utilizando o motor gráfico Unity3D (versão: 2021.3.23f1). Além disso, foi utilizado o ambiente de desenvolvimento Visual Studio Community 2022 (versão: 17.5.1), que é disponível de maneira gratuita.

As cenas do jogo que não possuem interação direta com o jogador são criadas utilizando dois componentes principais como base: o cenário da cena específica e a cinemática que será mostrada nessa cena. O cenário é montado utilizando modelos, sejam eles de construções, personagens, animais ou plantas, e todos os modelos utilizados no jogo foram adquiridos através da *asset store* que a própria Unity oferece. Quanto a parte da cinemática, é possível dividi-la em duas ferramentas essenciais disponibilizadas pelo próprio Unity: Cinemachine e Timeline.

Timeline é a ferramenta que o Unity disponibiliza para a criação de sequências de eventos (UNITY TECHNOLOGIES, 2020a). Essa ferramenta foi utilizada para criar as sequências de posicionamento, rotação e demais possibilidades de comportamentos de câmeras e objetos nas cinemáticas do jogo. Quanto à Cinemachine, Unity Technologies (2020b) a classifica como um sistema de câmeras que permite a utilização de comportamentos mais complexos de câmeras sem a necessidade de escrever código. Essa ferramenta foi utilizada em todas as câmeras das cinemáticas do jogo.

No jogo existem três tipos diferentes de Non-Player Characters (NPC), com os quais o jogador não interage diretamente, apenas acompanha. Desses três tipos de NPC, dois deles são controlados via código e o terceiro se move conforme o que foi determinado na cinemática através da ferramenta Timeline. Na Figura 16 é possível observar um exemplo do terceiro tipo de NPC citado, no qual o objeto `Fish (Animator)` tem sua posição e rotação alteradas conforme o tempo da cena avança.

Figura 16 - Exemplo de NPC controlado pela Timeline



Fonte: elaborado pelo autor.

Quanto aos outros dois tipos de NPCs, o mais simples deles é controlado por uma classe chamada `NPC`, a qual está demonstrada no Quadro 4. Essa classe determina os valores de animação desse NPC, sendo eles velocidade e se o NPC está encostado em uma parede ou não.

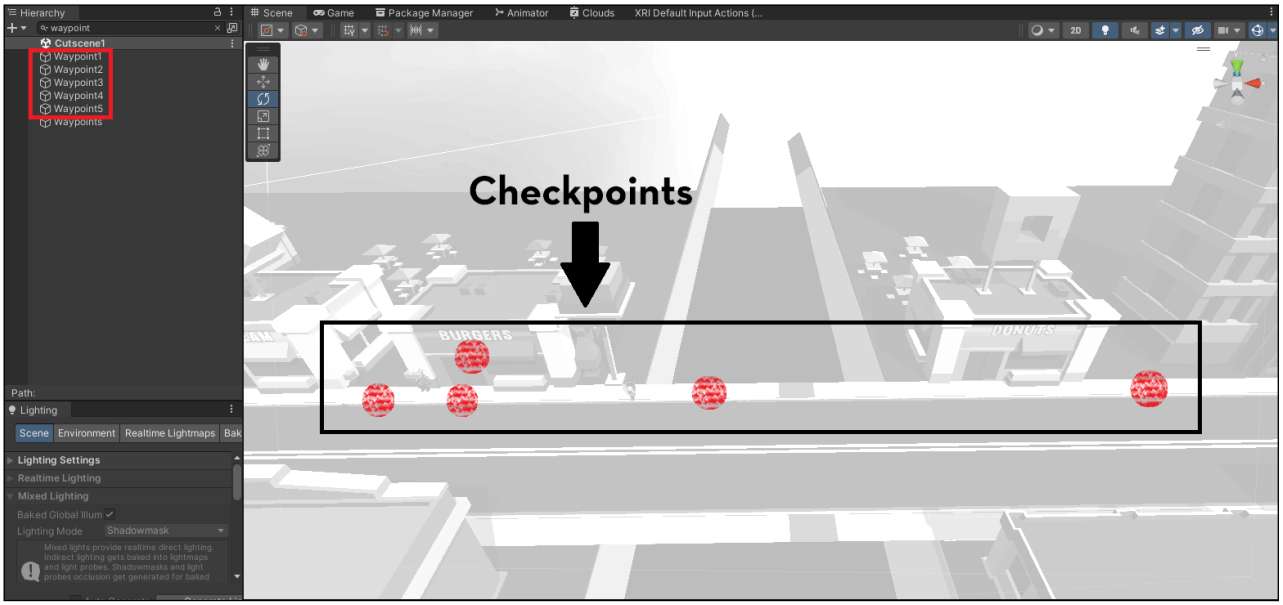
Quadro 4 - Classe `NPC`

```
Script do Unity (14 referências de ativo) | 0 referências
public class NPC : MonoBehaviour
{
    public float speed;
    public bool leaningInTheWall;

    private Animator animator;

    Mensagem do Unity | 0 referências
    void Start() {
        animator = GetComponent<Animator>();
    }

    Mensagem do Unity | 0 referências
    void Update() {
        if (animator != null) {
            animator.SetBool("Leaning_b", leaningInTheWall);
            animator.SetFloat("Speed_f", speed);
        } else {
            transform.Translate(Vector3.forward * Time.deltaTime * speed);
        }
    }
}
```

Fonte: elaborado pelo autor.

O último tipo de NPC é um pouco mais detalhado em suas ações e é também o menos utilizado. Esse NPC possui um código que serve para avançar até algum local específico da cena e passar por *checkpoints* durante o caminho. Além disso, seu código também permite a manipulação de um objeto que o NPC possa estar segurando. Para que ele possa exercer sua função corretamente, é necessário determinar alguns *checkpoints* no cenário e passá-los para o código. Os *checkpoints* são posicionados no cenário através de `GameObjects` com o nome "Waypoint" seguido de seu índice, conforme apresentado na Figura 17. Já a classe que controla esse tipo de NPC se chama `NPCMain`, e ela pode ser observada por completo no Quadro 5.

Figura 17 - Checkpoints da Cena 1, `Fase 2`



Fonte: elaborado pelo autor.

Quadro 5 - Classe `NPCMain`

```
Script do Unity (5 referências de ativo) | 0 referências
public class NPCMain : MonoBehaviour
{
    public Transform[] goals;
    private NavMeshAgent agent;
    private int currentGoal = 0;

    public Transform holdingObject = null;
    public int turnHoldingObjectVisibleAfterWaypoint = 0;
    public int droptObjectAtWaypoint = -1;
    public Camera initialCamera;

    private Animator animator;

    Mensagem do Unity | 0 referências
    void Start() {
        agent = GetComponent<NavMeshAgent>();
        animator = GetComponent<Animator>();
        agent.destination = goals[currentGoal].position;
    }

    Mensagem do Unity | 0 referências
    private void Update() {
        if (currentGoal < goals.Length) {
            agent.destination = goals[currentGoal].position;
            animator.SetFloat("Speed_f", agent.speed);

            if (!agent.pathPending && agent.remainingDistance < agent.speed) {
                currentGoal++;
            }

            CheckShowObject();
            CheckDropObject();
        } else {
            animator.SetFloat("Speed_f", 0);
        }
    }

    1 referência
    private void CheckShowObject() {
        if (currentGoal >= turnHoldingObjectVisibleAfterWaypoint && holdingObject != null) {
            if (holdingObject.TryGetComponent(out MeshRenderer holdingObjectRendered)) {
                holdingObjectRendered.enabled = true;
            }
        }
    }

    1 referência
    private void CheckDropObject() {
        if (currentGoal >= droptObjectAtWaypoint && holdingObject != null) {
            holdingObject.parent = null;
            Rigidbody rigidBody = holdingObject.GetComponent<Rigidbody>();
            rigidBody.useGravity = true;
        }
    }

}
```

Fonte: elaborado pelo autor.

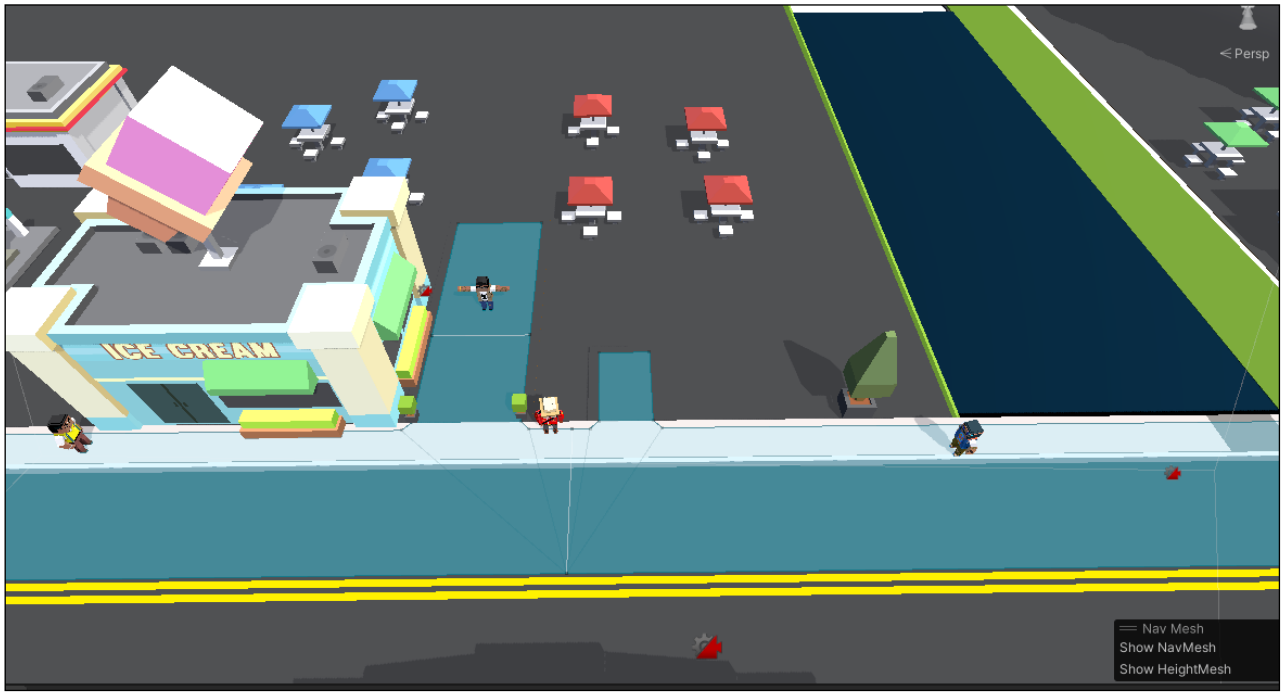
A classe `NPCMain` herda da classe `MonoBehaviour`, que é disponibilizada pelo Unity, permitindo assim que ela utilize alguns métodos da classe pai. Um desses métodos é o `Update`, que é executado a cada renderização realizada. O método `Update` está sendo utilizado nesse caso para poder determinar qual é o próximo destino para o qual o NPC deve seguir, bem como verificar se o objeto que está sendo segurado pelo NPC deve ser exibido ou desvinculado dele através dos métodos `CheckShowObject` e `CheckDropObject`.

Além dos *checkpoints*, que são recebidos através do atributo `goals`, a classe `NPCMain` também necessita de outros objetos para poder funcionar conforme o esperado. O atributo mais importante para isso é o `agent` que é atribuído no método `Start`, o qual também é herdado através da classe `MonoBehaviour`.

O atributo `agent` é do tipo `NavMeshAgent`, o qual faz parte da ferramenta `NavMesh` que o Unity disponibiliza. Essa ferramenta serve para determinar quais são as áreas navegáveis no cenário (Unity Technologies, 2023). Sua aplicação pode ser visualizada na Figura 18, aonde todos os objetos que estão em azul são aqueles sobre os quais os personagens

com a classe `NPCMain` podem andar. Para fins de demonstrar todos os locais, a figura não retrata a hamburgueria que existe ao lado da sorveteria.

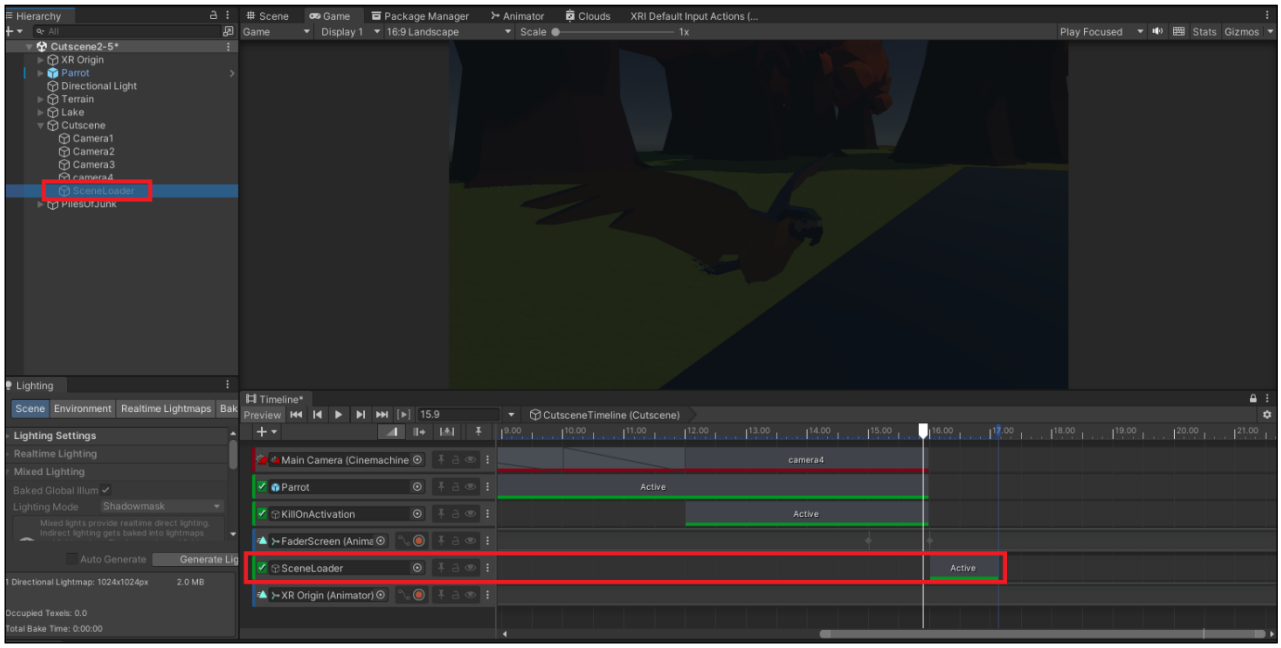
Figura 18 - Locais sobre os quais o NPC pode andar



Fonte: elaborado pelo autor.

Ao final de uma cinemática que leva a outra cinemática, a tela escurece e o objeto `SceneLoader` é ativado através da Timeline, conforme demonstrado na Figura 19. Esse objeto por sua vez possui um código vinculado a ele que utiliza o `SceneManager` do próprio Unity para carregar a próxima cena.

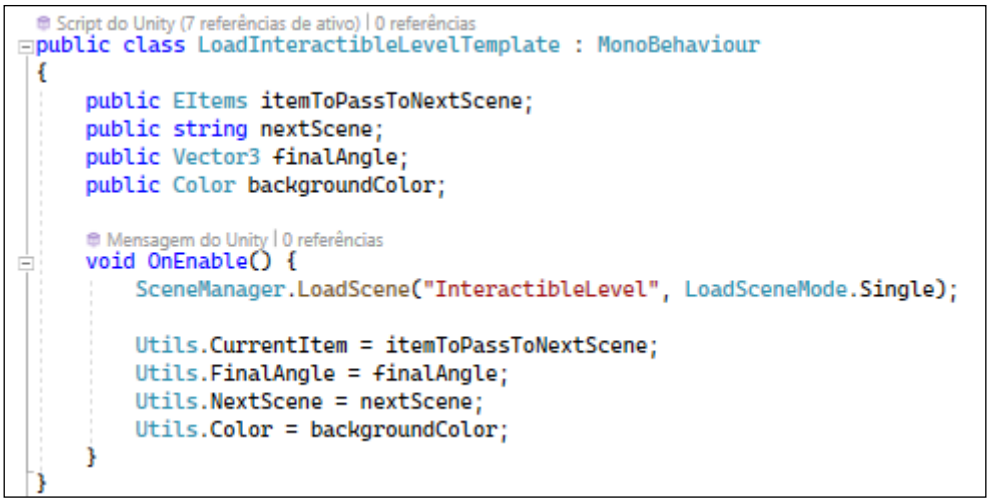
Figura 19 - Ativação do objeto SceneLoader



Fonte: elaborado pelo autor.

Nos casos em que a cinemática deve carregar não outra cinemática, mas sim a cena de interação, o código vinculado ao objeto `SceneLoader` é o `LoadInteractibleLevelTemplate`, que pode ser visualizado no Quadro 6. Esse código recebe todos os parâmetros necessários para carregar a cena de interação, sendo eles o item com o qual o jogador irá interagir, o ângulo final que o item deve estar para poder avançar de cena, qual é a cena para a qual o jogador será direcionado ao concluir a interação e a cor de fundo que deverá ser utilizada na interação.

Quadro 6 - Classe `LoadInteractibleLevelTemplate`

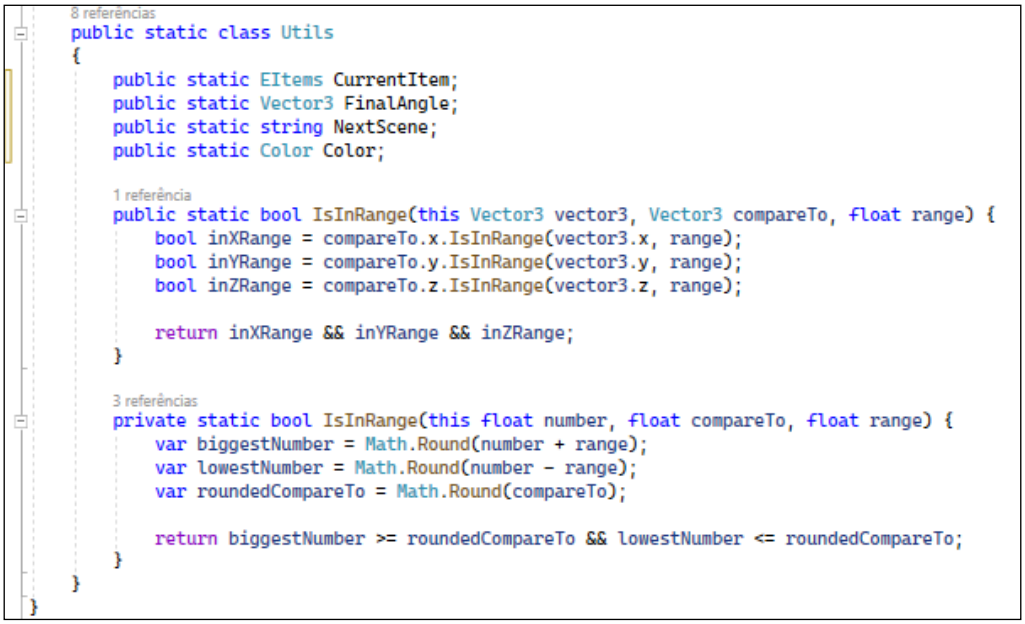


```
Script do Unity (7 referências de ativo) | 0 referências
public class LoadInteractibleLevelTemplate : MonoBehaviour
{
    public EItems itemToPassToNextScene;
    public string nextScene;
    public Vector3 finalAngle;
    public Color backgroundColor;

    Mensagem do Unity | 0 referências
    void OnEnable() {
        SceneManager.LoadScene("InteractibleLevel", LoadSceneMode.Single);

        Utils.CurrentItem = itemToPassToNextScene;
        Utils.FinalAngle = finalAngle;
        Utils.NextScene = nextScene;
        Utils.Color = backgroundColor;
    }
}
```

Fonte: elaborado pelo autor.

Conforme também é possível observar no Quadro 6, a classe `LoadInteractibleLevelTemplate` utiliza uma classe intermediária chamada `Utils` para determinar os parâmetros que serão passados para a cena de interação. Essa classe pode ser vista no Quadro 7, e é uma classe estática que tem duas utilidades principais. Sua primeira utilidade é justamente a passagem de parâmetros de maneira única entre as cenas cinemáticas e as de interação. A sua segunda funcionalidade é fornecer o método `IsInRange` para outras classes, que é utilizado para verificar se um vetor de três posições está dentro de um outro vetor de três posições, permitindo ter uma certa faixa de diferença aceitável para dizer que ambos possuem valores equivalentes. Esse método é aproveitado pela classe `InteractibleObject`, que o utiliza para verificar se o ângulo do objeto com o qual o jogador está interagindo é o ângulo desejado, permitindo assim a passagem do jogador para a próxima cena.

Quadro 7 - Classe `Utils`



```
8 referências
public static class Utils
{
    public static EItems CurrentItem;
    public static Vector3 FinalAngle;
    public static string NextScene;
    public static Color Color;

    1 referência
    public static bool IsInRange(this Vector3 vector3, Vector3 compareTo, float range) {
        bool inXRange = compareTo.x.IsInRange(vector3.x, range);
        bool inYRange = compareTo.y.IsInRange(vector3.y, range);
        bool inZRange = compareTo.z.IsInRange(vector3.z, range);

        return inXRange && inYRange && inZRange;
    }

    3 referências
    private static bool IsInRange(this float number, float compareTo, float range) {
        var biggestNumber = Math.Round(number + range);
        var lowestNumber = Math.Round(number - range);
        var roundedCompareTo = Math.Round(compareTo);

        return biggestNumber >= roundedCompareTo && lowestNumber <= roundedCompareTo;
    }
}
}
```

Fonte: elaborado pelo autor.

Por fim, a outra classe que utiliza a `Utils` é a `InteractibleObject`, que é a principal classe responsável pela interação do jogador com os objetos, e está vinculada a todos os objetos com os quais é possível interagir. O cenário de interação consiste em um cubo que pode ter suas paredes mudadas de cor. O jogador estará dentro desse cubo e visualizando um dos múltiplos objetos possíveis para realizar sua interação. A princípio, todos os objetos com os quais é possível interagir estarão ativos na cena, mas aqueles que não corresponderem ao objeto que está armazenado na classe `Utils` serão desativados durante a interação, conforme é demonstrado no Quadro 8.

Quadro 8 - Código que desativa o item caso ele não seja o objeto de interação

```
Mensagem do Unity | 0 referências
void Start() {
    inputData = GetComponentInParent<InputData>();
    rigidBody = GetComponent<Rigidbody>();

    isTheItem = Utils.CurrentItem == itemType;
    gameObject.SetActive(isTheItem);
}
```

Fonte: elaborado pelo autor.

Para permitir a interação com o jogador através dos óculos de realidade virtual, é necessário existir um jeito de receber as entradas que o jogador realizará, sejam elas apertos de botões ou movimentações dos controles. Para isso, o jogo utiliza o XR Interaction Toolkit, que de acordo com a Unity Technologies (2023), é um sistema de interação para criação de experiências tanto em realidade virtual quanto realidade aumentada.

Esse Toolkit oferece, entre vários outros, um componente chamado `XR Origin`, que é utilizado para colocar o jogador no mundo do jogo efetivamente, além de armazenar as diferentes interações possíveis com o jogador através dos óculos de realidade virtual. Consequentemente, existe a possibilidade de configurar manualmente cada possibilidade de interação, mas a ferramenta também disponibiliza um conjunto padrão de configurações que pode ser baixado a parte e utilizado automaticamente. Todas as configurações podem ser visualizadas na Figura 20 (Apêndice A).

Após as configurações terem sido finalizadas, estará quase tudo pronto para receber as entradas que o jogador fornece através dos seus óculos VR. O único ponto que falta é gerar um código que ficará responsável por, em cada cena em que a interação ocorre, rastrear se existem controles válidos para serem utilizados e armazená-los de uma maneira que fique de fácil acesso para as classes que quiserem utilizá-los. Esse processo pode ser visualizado no Quadro 9, através da classe `InputData`. Nessa classe, a cada renderização será realizada uma validação, para verificar se os objetos correspondentes aos controles ainda estão válidos. Caso eles não estiverem, os métodos `InitializeInputDevices` e `InitializeInputDevice` serão chamados para tentar instanciar corretamente os objetos.

Quadro 9 - Classe `InputData`

```
Script do Unity (1 referência de ativo) | 2 referências
public class InputData : MonoBehaviour
{
    public InputDevice _rightController;
    public InputDevice _leftController;
    public InputDevice _HMD;

    Mensagem do Unity | 0 referências
    void Update() {
        if (!_rightController.isValid || !_leftController.isValid || !_HMD.isValid)
            InitializeInputDevices();
    }

    1 referência
    private void InitializeInputDevices() {
        if (!_rightController.isValid)
            InitializeInputDevice(InputDeviceCharacteristics.Controller | InputDeviceCharacteristics.Right, ref _rightController);
        if (!_leftController.isValid)
            InitializeInputDevice(InputDeviceCharacteristics.Controller | InputDeviceCharacteristics.Left, ref _leftController);
        if (!_HMD.isValid)
            InitializeInputDevice(InputDeviceCharacteristics.HeadMounted, ref _HMD);
    }

    3 referências
    private void InitializeInputDevice(InputDeviceCharacteristics inputCharacteristics, ref InputDevice device) {
        List<InputDevice> devices = new List<InputDevice>();
        InputDevices.GetDevicesWithCharacteristics(inputCharacteristics, devices);

        if (devices.Count > 0) {
            device = devices[0];
        }
    }
}
```

Fonte: elaborado pelo autor.

Uma vez que a configuração é concluída e é possível acessar as entradas que o jogador fornece, e assim entra em cena mais uma vez a classe `InteractibleObject`. Conforme pode ser visualizado no Quadro 10, enquanto o jogador não interagir com o objeto, o objeto estará rodando sobre o próprio eixo. No momento em que o jogador começar sua interação, o método `RotateItem` faz com que o objeto receba a mesma rotação do controle do jogador. Por sua vez, o método `CheckIfTheAngleIsRight` realiza a validação da rotação do objeto, para ver se o jogador conseguiu chegar na rotação desejada e pode avançar para a próxima cena.

Quadro 10 - métodos de interação da classe `InteractableObject`

```
Mensagem do Unity | 0 referências
void Update() {
    if (!isTheItem) return;

    if (!startedInteraction) {
        transform.Rotate(
            rotationSpeed * Time.deltaTime * Vector3.right +
            rotationSpeed * Time.deltaTime * Vector3.up +
            rotationSpeed * Time.deltaTime * Vector3.forward,
            Space.Self
        );
    }

    RotateItem();

    CheckIfTheAngleIsRight();
}

1 referência
private void RotateItem() {
    if (!inputData._rightController.TryGetFeatureValue(UnityEngine.XR.CommonUsages.triggerButton, out bool isTriggered)) return;
    if (!inputData._rightController.TryGetFeatureValue(UnityEngine.XR.CommonUsages.devicePosition, out Vector3 controllerPosition)) return;
    if (!inputData._rightController.TryGetFeatureValue(UnityEngine.XR.CommonUsages.deviceRotation, out Quaternion controllerRotation)) return;

    if (!isTriggered) {
        initialPosition = controllerPosition;
        return;
    } else Debug.Log(transform.eulerAngles);

    var position = initialPosition - controllerPosition;

    if (position == Vector3.zero) {
        rigidBody.angularVelocity = Vector3.zero;
    }

    if (!startedInteraction)
        startedInteraction = true;

    transform.rotation = controllerRotation;
    return;
}

1 referência
private void CheckIfTheAngleIsRight() {
    if (transform.eulerAngles.IsInRange(Utils.FinalAngle, 5)) {
        SceneManager.LoadScene(Utils.NextScene, LoadSceneMode.Single);
    }
}
```

Fonte: elaborado pelo autor.

4 RESULTADOS

Nessa seção são apresentados os dois tipos de testes realizados com o jogo. Na primeira subseção são discutidos os testes de funcionalidades do jogo, realizados durante o processo de desenvolvimento. Já na segunda subseção são discutidos os resultados oriundos da pesquisa realizada com os usuários que testaram o jogo após o desenvolvimento.

4.1 TESTES DE FUNCIONALIDADE

Os testes de funcionalidade foram realizados constantemente durante o desenvolvimento do jogo, para garantir que as funcionalidades estivessem de acordo com o esperado. Esses testes foram realizados principalmente com a utilização do HMD Meta Quest 2. Parte dos testes também foram realizados diretamente no Unity, mas esses testes dificilmente representavam o que aconteceria ao rodar o jogo com o HMD, já que diferente do Unity, o HMD permite que o jogador olhe ao redor do cenário livremente.

Um dos pontos principais que faz com que testes sem o uso do HMD não representem o resultado corretamente é o fato de que a rotação da câmera utilizada durante a cinemática não representa necessariamente a rotação na qual o componente `XR Origin` utilizará. Isso pode fazer com que enquanto a câmera esteja focada no personagem principal da cena, o componente esteja virado, por exemplo, 180º no eixo `x`, perdendo assim todo o foco da cena, ou então obrigando o jogador a movimentar sua cabeça no mundo real. Devido a isso, era necessário continuamente realizar pequenos ajustes nas rotações desse componente junto das câmeras, para garantir que o jogador não perdesse o foco da história conforme ela avançasse.

Inicialmente, o objetivo era que o jogo fosse desenvolvido para celulares, de uma maneira que o jogador poderia utilizar um Google Cardboard, o que aumentaria muito a quantidade de pessoas que teriam acesso ao jogo, já que esses dispositivos são muito mais baratos do que um HMD imersivo. De qualquer maneira, se provou um grande desafio

facilitar a interação do usuário com o objeto, e devido a isso, a plataforma de desenvolvimento foi alterada para o HMD, no caso, o Meta Quest 2.

4.2 TESTES COM USUÁRIOS

Após o desenvolvimento, foram realizados testes com quatro usuários. Não houve um local específico para a aplicação dos testes. Todos os testes foram realizados com a presença do autor, utilizando o HMD dele. Durantes a aplicação dos testes, foi primeiramente apresentada a ideia do jogo, e em seguida os jogadores começavam a jogar. Após todos terem concluído seus testes, foi solicitado o preenchimento de um formulário, que está demonstrado no Quadro 11 junto das respostas dos participantes.

Quadro 11 - Respostas dos usuários

Pergunta	**Respostas**	**Quantidade**
Qual a sua faixa etária?	De 10 a 14 anos De 19 a 25 anos De 36 a 50 anos	25% 25% 50%
Você já havia utilizado um dispositivo de realidade virtual imersiva antes? (Oculus/Meta Quest ou parecidos)	Sim	100%
Numa escala de 0 a 10, quanto você gostou da sua experiência com o jogo Consequências?	7 9 10	25% 25% 50%
Quão difícil você achou o jogo (escala 1[muito fácil a 5 [muito difícil])?	2 3 4	50% 25% 25%
Você sentiu algum tipo de tontura ou desconforto devido a movimentações abruptas da câmera durante o jogo?	Não	100%
Foi possível entender a história de todas as fases?	Sim	100%
Caso não, por gentileza, descreva qual(is) história(s) não foi possível compreender e por quê.	-	
Na sua opinião, seria possível utilizar o jogo Consequências em escolas para ajudar na conscientização ambiental dos alunos?	Sim	100%
Caso não, por gentileza, descreva o motivo.	-	

Fonte: elaborado pelo autor.

A última pergunta foi aberta, para que os jogadores pudessem deixar seus comentários sobre o que acharam do jogo e o que poderia ser melhorado. Foram obtidas três respostas para essa questão. Duas delas foram positivas e comentaram como gostaram da história do jogo e que ele é bem educativo. Houve um terceiro e último comentário que pontuou algumas coisas a melhorar no jogo, sendo elas: adicionar sons ao jogo; facilitar a interação do jogador com os objetos; realizar uma melhor ambientação dos cenários subaquáticos; adicionar animações em interações dos NPCs, como por exemplo ao passar por portas.

Os resultados são majoritariamente satisfatórios. O objetivo de fazer com que os jogadores tivessem facilidade de compreender a história sendo narrada foi concluído com êxito, e o fato de todos terem concordado que o jogo seria viável em uma sala de aula também é positivo para o objetivo. Todavia, alguns jogadores tiveram uma grande dificuldade com a interação com os objetos no jogo. Essa dificuldade se deve ao fato de que o ângulo final ao qual o objeto deve chegar para que o jogador possa avançar de cena é muito específico, fazendo assim com que o jogador tenha interações onde ele consegue chegar no objetivo de transformar o objeto atual em outro, mas o ângulo está trinta graus maior no eixo y do que o esperado, por exemplo.

5 CONCLUSÕES

Com base nos resultados obtidos, o jogo alcançou seu objetivo de utilizar a RV para promover a conscientização ambiental. A incorporação do tema “conscientização ambiental” foi facilmente implementada e agregou muito valor ao resultado do jogo. Outro ponto relevante é que o especialista que ajudou no desenvolvimento dos roteiros comentou sobre como o jogo pode ser utilizado para ajudar na compreensão do tema por crianças. Por fim, houve alguns pontos que não foram desenvolvidos conforme o esperado, como o fato de o jogo não possuir sons, as cenas que ocorrem no fundo do mar não parecerem estar no fundo do mar, bem como dificuldade que existe para chegar no ângulo correto com os objetos durante as interações.

O Unity se mostrou uma ferramenta mais do que suficiente para o desenvolvimento em RV e atendeu todas as necessidades com excelência. Algumas das ferramentas utilizadas possuíam uma curva de aprendizado mais atenuada, mas a utilização da linguagem C# balanceou essa curva devido à utilização prévia e conhecimento dessa linguagem de programação. De qualquer maneira, a comunidade que existe ao redor do Unity facilitou o aprendizado e a correção dos

problemas que apresentaram maior dificuldade. A integração do Unity com o Meta Quest 2 também foi simples e direta, sem dificuldades e rápida.

Quanto às ferramentas utilizadas: O XR Interaction Toolkit foi um dos processos mais complicados de realizar as configurações, e isso acabou incorrendo em problemas durante a implementação, mas uma vez configurado corretamente, funcionou sem demais problemas. No começo, a Timeline e a Cinemachine foram duas ferramentas complicadas de utilizar de maneira eficaz. De qualquer forma, uma vez compreendidas, elas aceleraram o processo de desenvolvimento das cinemáticas das cenas, fazendo com que as últimas cenas criadas tomassem cerca de um vigésimo do tempo que a primeira cena levou.

Apesar do sucesso e alcance dos objetivos propostos, foram identificadas algumas possibilidades para continuação e extensão da pesquisa desse projeto:

a) aumentar a quantidade de fases;
b) aumentar a quantidade de objetos com os quais é possível interagir;
c) disponibilizar o jogo também para Google Cardboard, facilitando assim o acesso a ele;
d) aplicar o jogo em turmas do ensino fundamental e avaliar o impacto realizado;
e) adicionar música e efeitos sonoros no jogo;
f) facilitar a interação do jogador com os objetos.

REFERÊNCIAS

AZUMA, Ronald T. A survey of augmented reality. **Presence: Teleoperators & Virtual Environments**, Cambridge, v. 6, n. 4, p. 355-385, ago. 1997.

BRASIL. Ministério da Educação. Secretaria de Educação Básica. **Caderno meio ambiente**. Brasília, DF, 2022. Disponível em: http://basenacionalcomum.mec.gov.br/images/implementacao/cadernos_tematicos/caderno_meio_ambiente_consolidado_v_final_27092022.pdf. Acesso em: 24 jun. 2023.

DING, Y.; LI, Y.; CHENG, L.. Application of Internet of Things and Virtual Reality Technology in College Physical Education. **IEEE Access**, New York, v. 8 n. 1, p.96065-96074, maio 2020.

HOMBURG A., STOLBERG A. Explaining pro-environmental behavior with a cognitive theory of stress. **Journal of Environmental Psychology**, Amsterdan, v. 26, n. 1, p.1-14, mar. 2006.

IBISWORLD, **Global Movie Production & Distribution Industry** – Market Research Report. 2022. Disponível em: https://www.ibisworld.com/global/market-research-reports/global-movie-production-distribution-industry/. Acesso em: 25 jun. 2023.

META. **Get Meta Quest 2 for less**, 2023. Disponível em: https://www.meta.com/quest/products/quest-2. Acesso em: 25 jun. 2023.

NASCIMENTO, K. R. F; SANTOS, M. R. R.; SILVA, J. A. Sacolas Biodegradáveis: Sustentabilidade e ascensão da produção. **Diversitas Journal**, Arapiraca, v. 7, n. 1, p.171-189, jan./abr. 2022.

NEWZOO, **Newzoo Global Games Market Report 2022 | Free Version**. 2023. Disponível em: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2022-free-version. Acesso em: 24 jun. 2023.

NIENOW, Matheus N. **CidadaniaAR – Jogo de puzzle utilizando realidade aumentada com ilusão de ótica**. 2019. 24f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SÁNCHEZ, M.J.; LAFUENTE, Regina. Defining and measuring environmental consciousness. **Revista Internacional de Sociologia**, Madrid, v. 68, n. 3, p.731-735, set./dez. 2010.

SILVA R. W. C., PAULA B. L. Causa do aquecimento global: antropogênica versus natural. **Terræ Didatica**, São Paulo, v. 5 n. 1, p.42-49, jul. 2009.

SKELETON BUSINESS. **Vignettes**. 2017. Disponível em: https://vignettesga.me/. Acesso em: 18 set. 2022.

TAO, G. *et al*. Immersive virtual reality health games: a narrative review of game design. **Journal of NeuroEngineering and Rehabilitation**, v. 18, n. 31, p.1-21, fev. 2021.

UNITY TECHNOLOGIES, **Introduction to Timeline**, 2020a. Disponível em: https://learn.unity.com/tutorial/introduction-to-timeline-2019. Acesso em: 24 jun. 2023.

UNITY TECHNOLOGIES. **Overview of Cinemachine**, 2020b. Disponível em: https://learn.unity.com/tutorial/overview-of-cinemachine. Acesso em: 24 jun. 2023.

UNITY TECHNOLOGIES, **Working with NavMesh Agents**, 2021. Disponível em: https://learn.unity.com/tutorial/working-with-navmesh-agents. Acesso em: 24 jun. 2023.

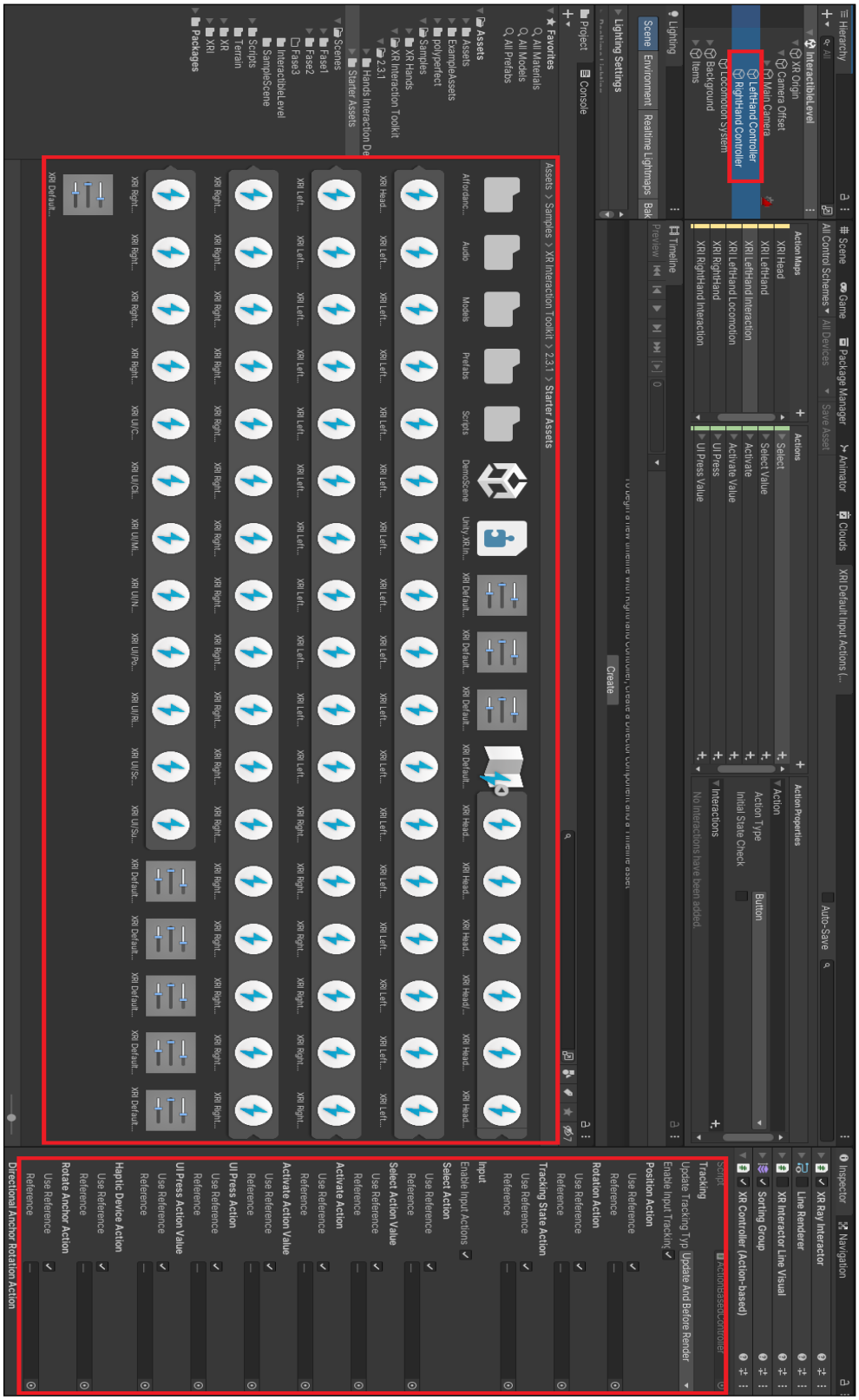
UNITY TECHNOLOGIES, **XR Interaction Toolkit**, 2023. Disponível em: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html. Acesso em: 24 jun. 2023.

USTWO GAMES. **Monument Valley**. 2014. Disponível em: https://www.monumentvalleygame.com/. Acesso em: 28 set. 2022.

APÊNDICE A – CONFIGURAÇÕES DISPONIBILIZADAS PELO XRINTERACTIONTOOLKIT

Nesse apêndice são apresentadas as configurações padrões que são disponibilizadas pelo XrInteractionToolkit. Conforme pode ser observado na Figura 20, estão destacados à esquerda superior os componentes aos quais as configurações devem ser anexadas, à esquerda inferior todos os arquivos que vêm com o pacote de Starter Assets que a ferramenta provê, e à direita os locais onde esses arquivos devem ser referenciados.

Figura 20 - Arquivos que a ferramenta XR Interaction Toolkit provê



Fonte: elaborado pelo autor.