# CIDADANIAAR - JOGO DE PUZZLE UTILIZANDO REALIDADE AUMENTADA COM ILUSÃO DE ÓTICA

**Matheus Navarro Nienow, Dalton Solano dos Reis** – Orientador

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

mnnienow@furb.br, dalton@furb.br

***Resumo:*** *Este artigo apresenta o processo de desenvolvimento e avaliação de um jogo para plataformas móveis em realidade aumentada utilizando ilusão de ótica. O objetivo é criar um jogo que faça uso da ilusão de ótica como mecânica principal de jogabilidade, incorporando o tema de reciclagem de forma interativa. O jogo foi desenvolvido utilizando o motor gráfico Unity, com a linguagem C# e o plugin para realidade aumentada Vuforia. Os dispositivos utilizados para o desenvolvimento e testes foram do tipo handheld displays. As ilusões de ótica utilizadas são do tipo paradoxo e da classe cognitiva. Para sua avaliação, foram realizados testes e aplicado um questionário com usuários. Os resultados obtidos foram satisfatórios, de forma que o jogo cumpriu seus objetivos.*

***Palavras-chave****: Ciência da computação. Realidade Aumentada. Jogo. Puzzle. Ilusão de ótica. Unity. Vuforia. Vetor. Enigma.*

## 1 INTRODUÇÃO

Atualmente, a tecnologia está diretamente ligada a diversos aspectos da vida das pessoas, incluindo o lazer e a educação. Em 2018, os videogames geraram 137,9 bilhões de dólares (NEWZOO, 2019) enquanto o cinema gerou 136 bilhões de dólares (IBISWORLD, 2018) e a música gerou 19,1 bilhões de dólares (IFIP, 2019). Com base nisso pode-se afirmar que a indústria de videogames é a mais lucrativa dentre as indústrias do entretenimento. Dos 137,9 bilhões de dólares, 51% são decorrentes de jogos para tablets e smartphones, o que mostra o tamanho e potencial do mercado móvel (NEWZOO, 2019).

A Realidade Aumentada (RA) é uma tecnologia derivada da Realidade Virtual (RV) que ao invés de substituir o mundo real por um virtual, mistura ambos, sobrepondo objetos virtuais ao mundo real (AZUMA, 1997). Essa tecnologia está fortemente ligada ao mercado de dispositivos móveis pela praticidade de utilização de aplicações do tipo e permite que jogos inovadores sejam criados, possibilitando diferentes formas de interação com as ferramentas. Isso abre um leque de possibilidades para a integração da realidade aumentada na educação.

Segundo Newzoo (2019), "jogos" é a segunda maior categoria de aplicativos de RA da plataforma iOS, com 17% do total de aplicativos, estando atrás apenas da categoria "entretenimento", com 20% do total de aplicativos da plataforma em 2018. Isso mostra a popularidade e alcance dos games no mercado de RA.

Nos últimos anos houve um aumento considerável na quantidade de ferramentas e jogos utilizando RA criados para auxiliar na educação, instigando a curiosidade, lógica e outros fatores. Ainda assim, é preciso buscar inovações na jogabilidade e nas mecânicas empregadas nestes jogos para atrair e manter a atenção do jogador e para que o instigue a pensar de formas inusitadas na resolução dos problemas. Com base nisto, este trabalho tem por objetivo construir um jogo para plataformas móveis com Realidade Aumentada que faça uso de mecânicas de jogabilidade baseadas na ilusão de ótica. Os objetivos específicos são: utilizar ilusão de ótica como mecânica principal do jogo; criar um sistema de interação com o jogo utilizando elementos do mundo real ao invés de menus na tela; incorporar o tema de reciclagem no jogo.

## 2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os assuntos que fundamentam o desenvolvimento do jogo. A primeira seção trata sobre a realidade aumentada, a segunda seção trata sobre a ilusão de ótica e a terceira seção trata sobre os trabalhos correlatos.

### 2.1 REALIDADE AUMENTADA

Segundo Azuma (1997, p. 2) e Zhou, Duh e Billinghurst (2008, p. 193), a RA complementa a realidade ao invés de substitui-la, como é o caso da realidade virtual. A RA permite ao usuário enxergar a realidade com objetos virtuais sobrepostos, misturando o mundo real com o mundo virtual. Azuma (1997, p. 2) define RA como um sistema que combine o mundo real com o mundo virtual, seja interativo em tempo real e registrado em 3D. Isso garante que outras aplicações não sejam caracterizadas como realidade aumentada, como por exemplo filmes que possuem objetos virtuais misturados ao mundo real, porém não são interativos.

Existem três tipos de monitores para realidade aumentada: *Head Mounted Displays* (HMD), *handheld displays* e *spatial displays* (CARMIGNIANI, 2011). Segundo Carmigniani (2011) e Zhou, Duh e Billinghurst (2008, p. 197), o HMD é um display que é utilizado na cabeça, com um capacete ou um par de óculos, e pode ser do tipo *video-see-through* ou *optical-see-through*. O HMD do tipo *video-see-through* utiliza uma câmera montada no display para que o usuário consiga enxergar o ambiente ao seu redor através de um *stream* de vídeo diretamente no display, isso permite que a aplicação tenha um alto nível de sincronismo entre os objetos virtuais e os objetos do ambiente real, já que o próprio HMD exibe a imagem do mundo real. Porém, esse aparelho tem um processamento muito mais alto por conta da necessidade do fazer o *stream* de vídeo. Já o HMD *optical-see-through* utiliza um display translúcido e apenas reproduz os objetos virtuais no display, enquanto a imagem do ambiente real passa através do display translúcido até os olhos do usuário, isso reduz o processamento, porém faz com que seja difícil ter uma sincronia perfeita entre os objetos virtuais e o mundo real.

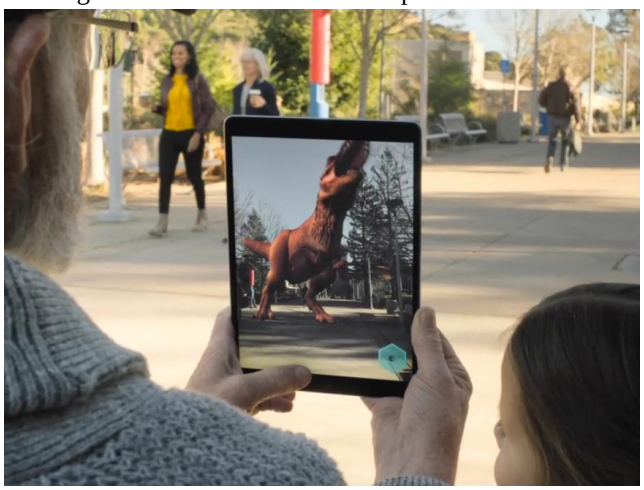
A Figura 1 apresenta o Microsoft HoloLens 2, um HMD que utiliza a técnica *optical-see-through*. O dispositivo é um capacete que apresenta um display translúcido e faz uso de diversos sensores para capturar o posicionamento da cabeça do usuário e então sobrepor objetos virtuais através do display.

Figura 1 - Microsoft HoloLens 2

Fonte: Microsoft (2019).

*Handheld displays* são dispositivos pequenos que podem ser segurados nas mãos pelo usuário, como smartphones e tablets. Esses dispositivos utilizam câmeras para registrar o ambiente real e fazem a projeção dos objetos virtuais sobre as imagens diretamente no display do dispositivo. Além de possuírem diversos sensores como GPS, giroscópio, acelerômetro, que permitem uma maior precisão no posicionamento dos objetos virtuais no ambiente (CARMIGNIANI, 2011). Azuma et al (2001) descreve *handheld displays* como “lentes de aumento” que mostram o mundo real com uma camada de RA. A Figura 2 mostra um iPad, um tablet da Apple, executando uma aplicação em RA onde um dinossauro virtual é sobreposto sobre o mundo real através do display, câmera e sensores do dispositivo. Zhou, Duh e Billinghurst (2008, p. 198) indicam que *handheld displays* são ótimas alternativas a HDM por serem pouco intrusivos, de fácil acesso, extremamente móveis e socialmente aceitáveis.

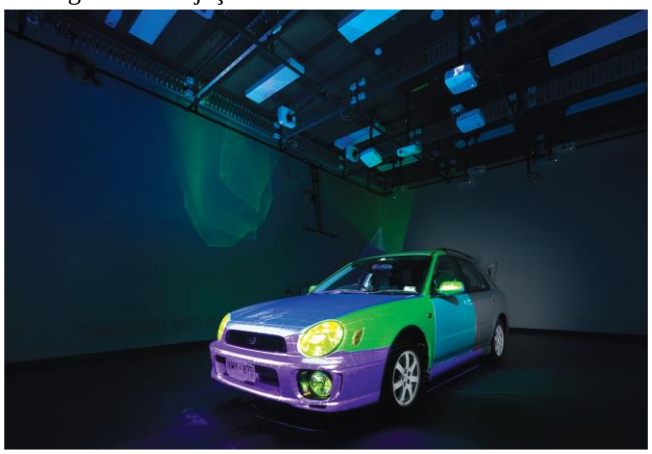
Figura 2 - iPad executando um aplicativo com RA



Fonte: Apple (2019).

*Spatial Augmented Reality* (SAR) ou Realidade Aumentada Espacial numa tradução livre, segundo Carmigniani (2011), consiste na projeção de objetos virtuais diretamente no espaço físico através de projetores, sem a necessidade de vestir ou segurar os dispositivos. Azuma et al (2001) chama essa modalidade de *Projections Displays*. Na Figura 3 é possível visualizar uma aplicação da SAR, onde peças virtuais são projetadas sobre um veículo real.

Figura 3 - Projeção com SAR feita sobre um veículo



Fonte: Marner (2014).

## 2.2 ILUSÃO DE ÓTICA

Gregory (1991) categoriza as ilusões de ótica em três classes: físicas, psicológicas e cognitivas. Além disso, ele também indica quatro tipos de ilusão de ótica: ambiguidades, distorções, paradoxos e ficções. O Quadro 1 apresenta exemplos de ilusões de ótica para cada tipo e classe.

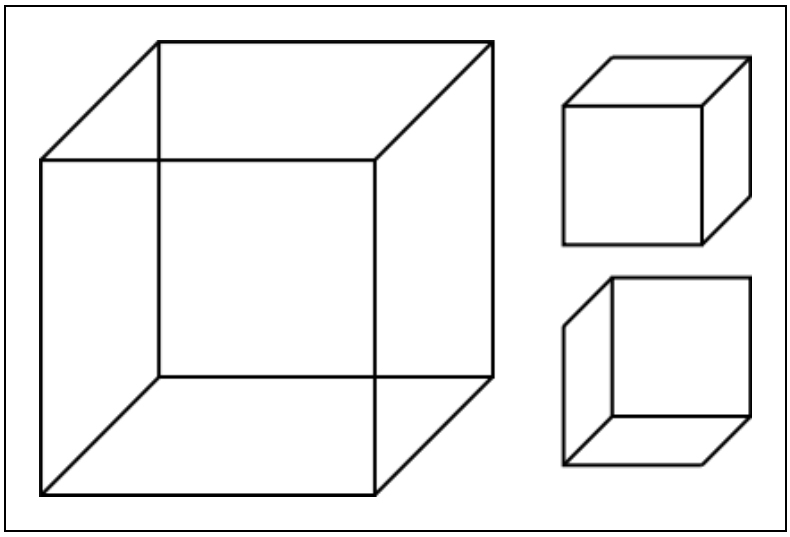
Quadro 1 - Classificações de ilusões de ótica

| | Física | Psicológica | Cognitiva |
|---|---|---|---|
| Ambiguidades | Sombras da névoa | Tamanho – tamanho para um olho estacionário<br>Real – movimento aparente | Cubo de Necker<br>Vaso de Rubin<br>Coelho-pato de Jastrow |
| Distorções | (de espaço) Galho na água<br>(de velocidade) Estroboscópio<br>(de cor) Filtros, refração, difração, espalhamento | (de espaço) Adaptações a tamanho, inclinação ou curvatura<br>Parede de café<br>(de brilho e cor) Contraste simultâneo e sequencial | Ponzo<br>Poggendorff<br>Orbison<br>Hering<br>Müller-Lyer<br>Figuras de Zöllner |
| Paradoxos | Espelhos (ver a si mesmo no lugar errado, duplicado) | Quando os canais visuais discordam<br>Efeito posterior do movimento: movimento ainda não mudando de posição<br>ou tamanho | Objetos impossíveis de Penrose<br>Imagens de Escher |
| Ficções | Arco-íris<br>Padrão moiré | Pós-imagens<br>Efeito autocinético<br>Padrões de migraine | Triângulo de Kaniza<br>Preenchimento de ponto cedo e escotoma |

Fonte: Gregory (1991).

O Cubo de Necker é um desenho bidimensional que representa um cubo tridimensional através das arestas do cubo. Ele é apresentado como uma ilusão de ambiguidade cognitiva, pois é possível perceber o cubo de duas maneiras. A primeira percepção indica que a face mais próxima do cubo se encontra no canto inferior esquerdo da figura. Porém, também é possível interpretar que a face mais próxima do cubo está no canto direito superior da figura. A Figura 4 apresenta o cubo original e suas duas formas de interpretação.

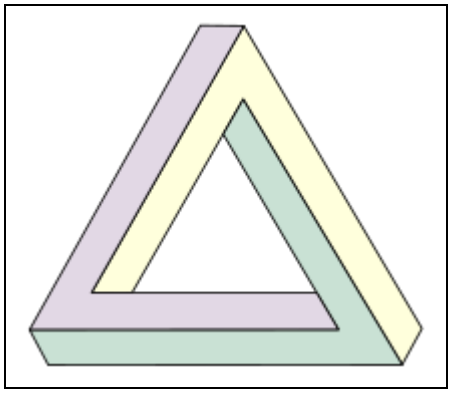
Figura 4 - Cubo de Necker



Fonte: Clair (2019).

Reutersvard criou o triângulo impossível em 1934 (PASHLEY, 2014), um desenho bidimensional que representa um triângulo tridimensional de características impossíveis. As três arestas se conectam de maneira que vistas como um todo representa um objeto impossível de ser reproduzido no mundo real, como mostra a Figura 5. Segundo Pashley (2014), essa ilustração foi mais tarde popularizada por Lionel Penrose em 1950, ficando então conhecido como o Triângulo de Penrose.

Figura 5 - Triângulo de Penrose



Fonte: Bromskloss (2019).

## 2.3 TRABALHOS CORRELATOS

São apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro está detalhado no Quadro 2, se trata de um jogo de puzzle 3D com ilusão de ótica, chamado Monument Valley (USTWO GAMES, 2014). O segundo trabalho é apresentado no Quadro 3, é um jogo desenvolvido em Realidade Aumentada para investigar o potencial da tecnologia no auxílio de ensino da Geografia (SILVA et al., 2014). O Quadro 4 descreve o terceiro trabalho, que consiste num jogo de realidade aumentada com o objetivo de auxiliar no ensino de sólidos geométricos para as séries iniciais (LEITÃO, 2013).

Quadro 2 - Monument Valley

| | |
|---|---|
| Referência | Ustwo Games (2014) |
| Objetivos | Produto comercial. |
| Principais funcionalidades | Resolução de enigmas utilizando ilusão de ótica. |
| Ferramentas de desenvolvimento | Motor gráfico Unity3D. |
| Resultados e conclusões | Monument Valley recebeu nota 89 no site Metacritic e foi uma das escolhas dos editores da App Store, recebeu o prêmio Apple Design Award (VITICCI, 2014), nomeado como o melhor jogo para iPad de 2014 (TACH, 2014), entre outros prêmios. |

Fonte: elaborado pelo autor.

O estúdio Ustwo Games (2014) utilizou Unity para desenvolver o jogo para as plataformas Android, iOS e Windows Phone. A ilusão de ótica é ponto chave da jogabilidade e o objetivo do jogador é levar o personagem principal do início até o final do mapa. Para tal, deve resolver enigmas criando caminhos formando objetos impossíveis, fazendo uso de diferentes estruturas que podem ser rotacionadas e movidas por todo o mapa, conforme demonstra a Figura 6.

Figura 6 - Fase inicial do jogo onde um caminho impossível é criado


Fonte: elaborado pelo autor.

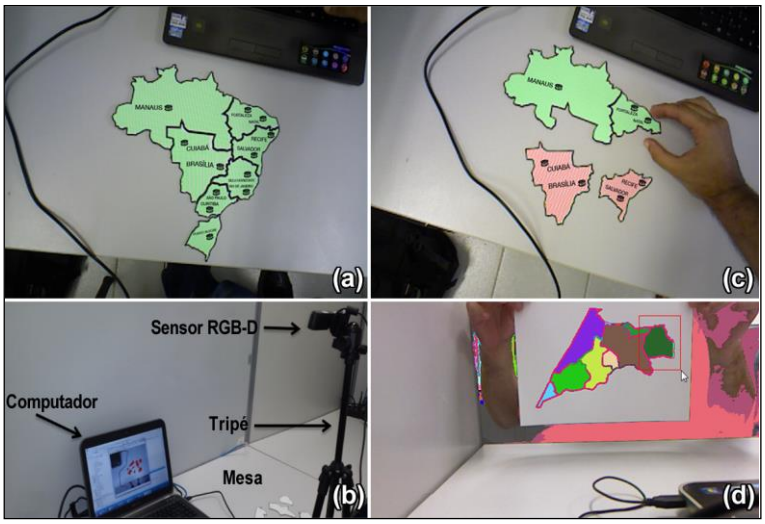
Quadro 3 - AR Jigsaw Puzzle

| | |
|---|---|
| Referência | Silva (2014) |
| Objetivos | O jogo busca investigar o potencial de utilização da RA no ensino de Geografia. |
| Principais funcionalidades | Resolução de quebra cabeça, adicionando texto, texturas e cores às peças para disponibilizar informações ao jogador. |
| Ferramentas de desenvolvimento | Não especificado. |
| Resultados e conclusões | Após a aplicação de questionários com especialistas, concluiu-se que o uso da RA em sala de aula pode trazer diversos benefícios. A ferramenta foi dada como atrativa e foi indicado a suma importância do trabalho em conjunto do professor com a ferramenta. |

Fonte: Silva (2014).

O jogo consiste num quebra cabeça onde informações, texturas e cores são adicionadas as peças físicas através da realidade aumentada em tempo de execução. A Figura 7 mostra um quebra cabeça das regiões do Brasil que contém os estádios utilizados na Copa do Mundo FIFA de 2014. O aplicativo está preparado para receber qualquer jogo do tipo quebra-cabeças ou montar, como o Tangram. É possível adicionar imagens, sons e vídeos sobre as peças, criando uma experiência completa no jogo.

Figura 7 - Capturas de tela do jogo




Fonte: Silva et al (2014).

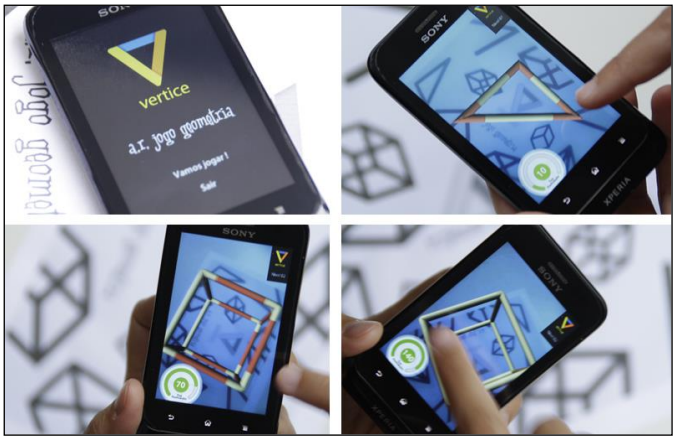
Quadro 4 - Vértice

| | |
|---|---|
| Referência | Leitão (2013) |
| Objetivos | Auxiliar no ensino de sólidos geométricos para séries iniciais. |
| Principais funcionalidades | Apresentação de imagens de sólidos geométricos e instruções por áudio, além de *feedback* auditivo. O jogo apresenta progressão de dificuldade. |
| Ferramentas de desenvolvimento | Unity3D em conjunto ao plugin de RA Vuforia. |
| Resultados e conclusões | Após testes comparativos, Leitão (2013) concluiu que o jogo melhora os índices de aprendizados se comparado aos métodos tradicionais de ensino como explicação verbal e ilustrações no quadro. Além disso, o fato de o jogo ser diferente do tradicional, ou seja, usar a realidade aumentada, desperta interesse, curiosidade e motivação no aluno. |

Fonte: Leitão (2013).

O jogo apresenta sólidos geométricos e dá instruções por áudio ao jogador, induzindo o aluno a tocar nas partes do sólido, conforme demonstra a Figura 8. Existem três níveis, onde são explorados os conceitos de vértices, arestas e faces, respectivamente. Segundo Leitão, para validar a efetividade do jogo no auxílio do ensino foi realizado um teste com duas turmas do ensino fundamental de uma escola de educação básica da cidade de Matosinhos (Portugal). Na primeira turma, foi feita uma explicação verbal e com ilustrações no quadro sobre o conteúdo (vértices, arestas e faces), já na segunda turma foi apresentado o jogo e todas as crianças puderam jogar uma de cada vez. Em ambas turmas foi aplicado um teste para mensurar os conhecimentos adquiridos pelos alunos. O teste consistia em três questões, cada uma apresentando um sólido geométrico com uma aresta, vértice ou face destacada e com três opções para identificar o elemento destacado em aresta, vértice ou face. Na primeira turma, de 27 alunos, 9 alunos identificaram a aresta, 15 alunos identificaram o vértice e 16 alunos identificaram a face. Já na segunda turma, de 26 alunos, 22 alunos identificaram a aresta, 23 alunos identificaram o vértice e 23 alunos identificaram a face (LEITÃO, 2013).

Figura 8 - Imagens do jogo em execução




Fonte: Leitão (2013).

## 3 DESCRIÇÃO DO JOGO

Nesta seção são apresentados os aspectos fundamentais e mais importantes da implementação do jogo. A primeira parte descreve a visão geral do jogo enquanto a segunda parte descreve a implementação do jogo.

### 3.1 VISÃO GERAL DO JOGO

O jogo desenvolvido coloca o jogador como observador do personagem principal e seu mundo. Existem três níveis no jogo, sendo o primeiro um nível tutorial e os dois seguintes desafios. O jogador interage com o mundo utilizando apenas a câmera do celular ou tablet. O jogo faz uso de um marcador que representa o cenário completo, esse marcador é reconhecido pela câmera do dispositivo e então é feita a projeção 3D do mundo virtual sobre o mundo real.

Em cada nível o jogador é apresentado com um cenário onde ele precisa ajudar o personagem a resolver algum desafio utilizando a câmera do dispositivo para criar uma perspectiva específica sobre objetos no cenário. O personagem se move automaticamente sobre o cenário e possui *checkpoints* específicos onde o personagem aguarda a ação do jogador para poder prosseguir. A Figura 9 exibe o primeiro nível, que serve como treinamento e tutorial para o jogador. Esse nível apresenta um cenário onde existem duas ilhas e uma ponte fora de posição, impossibilitando a conexão entre as ilhas. O personagem deve atravessar a cidade para chegar na loja vermelha do cenário, e então retornar para casa.

Figura 9 - Capturas de tela da fase de treinamento


Fonte: elaborado pelo autor.

O jogador deve utilizar a câmera para visualizar ponte e a cidade numa perspectiva que crie a ilusão de que a ponte está conectando as duas ilhas, o que permite a passagem do personagem. O jogo possui uma função de zoom, que é ativado por um toque duplo em qualquer parte da tela do dispositivo, o zoom permanece ativo por 5 segundos e então é desativado automaticamente. A Figura 10 demonstra a criação dessa ilusão pela perspectiva da câmera. Após isso, o jogador deve visualizar o letreiro na praia buscando formar o nome do jogo, CidadaniaAR, o que permite a passagem do personagem até a loja. Então, o personagem precisa voltar para casa e para isto o jogador deve criar a ilusão da ponte novamente para que o personagem possa voltar para a ilha inicial.

Figura 10 – Perspectiva conecta as duas ilhas pela ponte


Fonte: elaborado pelo autor.

A fase de treinamento faz uso dos dois tipos de puzzle que existem no jogo. O primeiro é o puzzle da ponte, que está sempre ativo (*Always On Puzzle*), toda vez que o personagem precisa passar por esse caminho, o jogador deve resolvê-lo. Já o segundo puzzle, do letreiro, funciona como um desbloqueio (*One Time Puzzle*), basta resolvê-lo uma vez que ele fica liberado até o fim do nível. No caso do nível de treinamento, uma vez que o jogador visualiza o letreiro, o caminho é liberado e não é mais necessário visualizá-lo para que o personagem passe por aquele caminho.

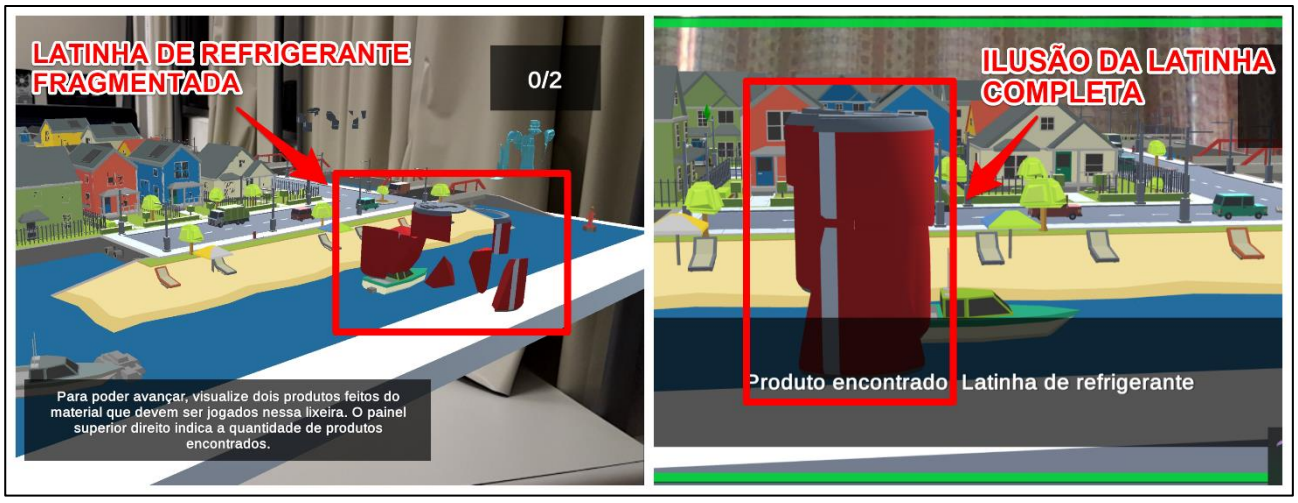
O segundo nível faz uso exclusivo de puzzles do tipo *One Time Puzzle*. Nesse nível o personagem está em outra parte da cidade e o jogador deve ajudar o personagem a encontrar os produtos corretos para serem reciclados nas lixeiras amarela (metal) e vermelha (plásticos). O cenário está exibido na Figura 11.

Figura 11 - Cenário do segundo nível


Fonte: elaborado pelo autor.

Existem quatro produtos espalhados ao redor do mapa, dois produtos feitos de plástico e dois de metal. Esses produtos estão fragmentados de forma que seja possível ver o objeto completo apenas de um ângulo específico, conforme demonstra a Figura 12. O objetivo do jogador é encontrar os produtos visualizando-os no ângulo correto por dois segundos para criar a ilusão de que esses objetos estão completos. No canto superior direito da tela é exibido um painel com a pontual do jogador. É possível fazer dois pontos para cada tipo de material. Primeiramente o jogador deve encontrar os produtos de metal e após isso deve encontrar os produtos de plástico. Após encontrar todos os produtos o jogador pode seguir para o terceiro nível.

Figura 12 - Latinha de refrigerante




Fonte: elaborado pelo autor.

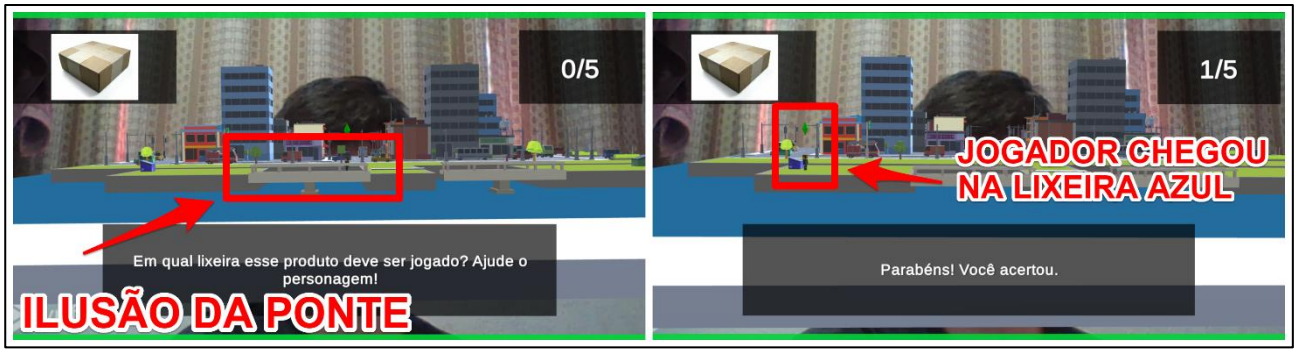
No terceiro nível o jogador deve continuar ajudando o personagem na reciclagem de lixo. Dessa vez o jogo apresentará imagens de produtos de papel e de vidro, e o jogador deverá resolver puzzles do tipo *Always On Puzzle*, no caso, o puzzle da ponte, para indicar em qual lixo esses produtos devem ser jogados. A Figura X apresenta o nível e seus elementos de tela. O item número 1 é o painel onde a imagem do produto é mostrada. O item 2 é o painel de pontuação do nível. Os itens 3 e 4 são as lixeiras de papel (azul) e vidro (verde), respectivamente. Os itens 5 e 6 são as pontes que permitirão ao personagem alcançar essas lixeiras após a visualização da ilusão pelo jogador.

Figura 13 - Terceiro nível do jogo


Fonte: elaborado pelo autor.

Para indicar em qual lixeira o produto é jogado o jogador deve visualizar a ilusão na ponte que completará o caminho até a lixeira, conforme demonstrado na Figura 14. Caso o jogador acerte, uma mensagem indicando o acerto será apresentada e a pontuação será incrementada. Após isso, o jogador deverá visualizar a ilusão novamente para que o personagem possa retornar até a ilha principal da cidade. Caso o jogador erre, uma mensagem indicando o erro será apresentada e a pontuação não será incrementada. Ao retornar a ilha principal uma nova imagem será carregada e o jogador deverá repetir esses passos. O nível consiste em 5 imagens. Após a finalização desse nível o jogo exibe um botão para que o jogador retorno ao menu inicial do jogo.

Figura 14 - Telas do terceiro nível do jogo




Fonte: elaborado pelo autor.

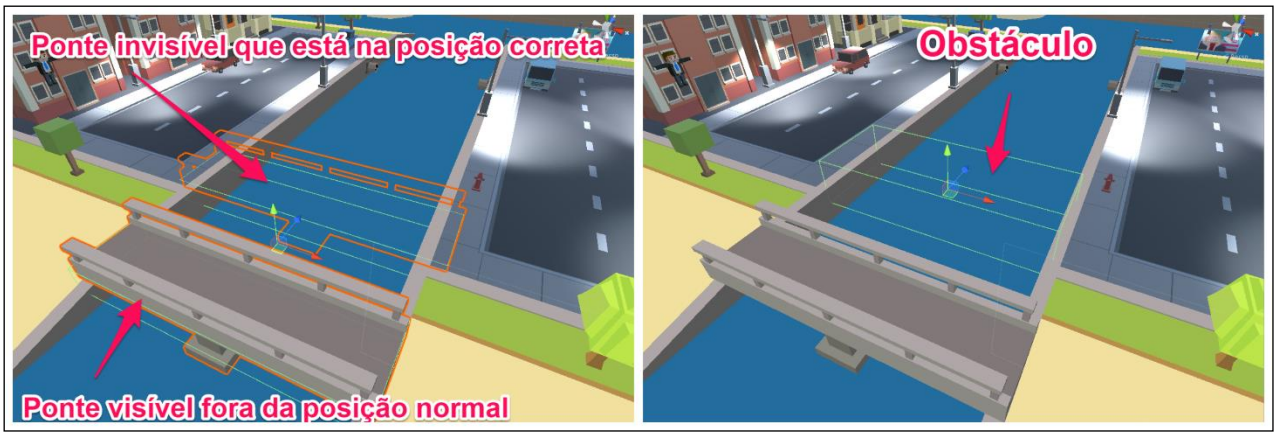
## 3.2 IMPLEMENTAÇÃO

O jogo foi desenvolvido com o motor gráfico Unity3D associado ao plugin para realidade aumentada Vuforia. O ambiente de desenvolvimento utilizado foi JetBrains Rider, uma ferramenta paga que possui uma licença educacional. Os cenários foram construídos utilizando os modelos de um pacote de *assets*, que provê diversas "peças" que se encaixam para formar diferentes partes de uma cidade.

Houve a necessidade de modificar alguns dos modelos utilizados no jogo e para isso foi utilizado a ferramenta de modelagem Blender. Muitos dos puzzles do jogo utilizam uma ilusão de perspectiva onde um objeto fragmentado é visto por completo apenas de um ângulo específico. Essa ilusão é explorada principalmente no segundo nível do jogo. O processo de fragmentação dos modelos está exibido passo a passo no Apêndice A.

Na Figura 15 é possível visualizar como a ilusão da ponte no treinamento inicial é criada. Utilizando duas pontes, uma fora do caminho e outra ponte invisível no lugar em que a ponte visível deveria estar. Há um `GameObject` que possui o componente `NavMeshObstacle` que cria um obstáculo na área de movimentação do personagem e o impede de passar pela ponte.

Figura 15 – Pontes e obstáculo na cena do nível de treinamento




Fonte: elaborado pelo autor.

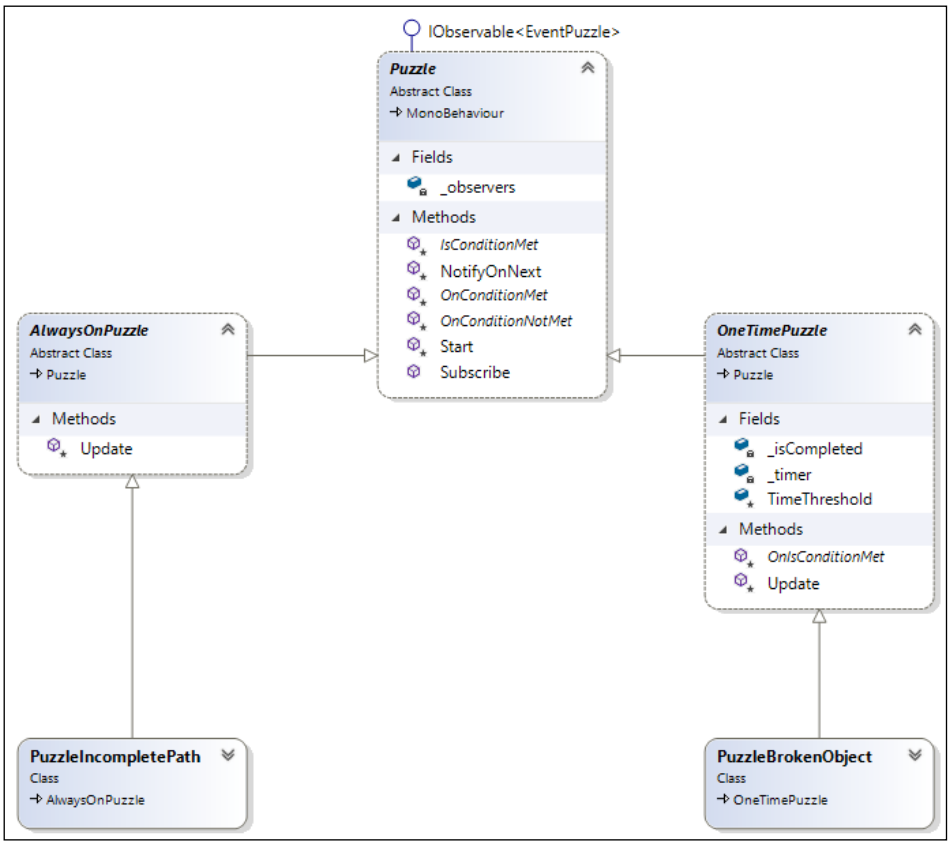
Quando o puzzle é resolvido esse obstáculo é desativado e o personagem consegue passar pela ponte invisível, porém, por conta da perspectiva é criada a ilusão de que ele está passando pela ponte que está fora do caminho. A Figura 16 exibe a ilusão nas visualizações da cena e da câmera do jogo.

Figura 16 - Visualização do puzzle na perspectiva da cena e do jogo


Fonte: elaborado pelo autor.

A estrutura de classes criada para os puzzles utilizados no jogo está exibida na Figura 17. Há uma classe abstrata `Puzzle` que define os métodos abstratos `IsConditionMet`, `OnConditionMet` e `OnConditionNotMet`. Esses métodos devem ser implementados por todos os tipos de puzzle dos jogos e definem como todos os puzzles devem ser criados. Essa classe implementa a interface `IObservable` e define alguns métodos e comportamentos padrões relacionados a essa interface, fazendo com que todos os puzzles possam enviar eventos para o resto do jogo utilizando o padrão de projeto *Observer*. Existem mais duas classes abstratas, `AlwaysOnPuzzle` e `OneTimePuzzle`, ambas herdando da classe `Puzzle`, que definem o comportamento dos dois tipos de puzzles no jogo.

Figura 17 - Diagrama de classes dos puzzles




Fonte: elaborado pelo autor.

A classe `AlwaysOnPuzzle`, exibida no Quadro 5, define o comportamento do puzzle que está sempre ativo no jogo. Esse comportamento verifica o método `IsConditionMet` nas linhas 7 e 8 e então de acordo com seu resultado executa o método `OnConditionMet` ou `OnConditionNotMet` nas linhas 10 e 13, respectivamente. Esses três métodos são herdados da classe pai `Puzzle`, e por permanecerem abstratos, não definem a implementação dos métodos, apenas a ordem e lógica de suas execuções. Isso permite que diversos puzzles sejam criados seguindo essa estratégia de funcionamento, porém com implementações de verificação e execução diferentes.

Quadro 5 - Classe abstrata `AlwaysOnPuzzle`

```
3   public abstract class AlwaysOnPuzzle : Puzzle
4   {
        Event function   Matheus Navarro Nienow
5       protected void Update()
6       {
7           var isResolved = IsConditionMet();
8           if (isResolved)
9           {
10              OnConditionMet();
11          } else
12          {
13              OnConditionNotMet();
14          }
15      }
16  }
```

Fonte: elaborado pelo autor.

O Quadro 6 exibe o código da classe `OneTimePuzzle`, que define o comportamento do puzzle que é resolvido apenas uma vez. Diferente da anterior, essa classe só executará o método `OnConditionMet` quando o método `IsConditionMet` retornar `true` por um tempo determinado. Na linha 20 é verificado a resolução do puzzle, caso essa verificação resulte em `false`, nas linhas 21 a 26 o temporizador é resetado. Caso o resultado da verificação seja true, incrementa-se o temporizador na linha 28 e é feita a verificação do tempo de resolução na linha 30. Além dos três métodos abstratos herdados, esse tipo de puzzle contém um método adicional chamado `OnIsConditionMet`, que é executado na

linha 29 enquanto as condições do puzzle estão sendo atendidas sem ter chegado ao tempo necessário para finalizar o puzzle. Isso permite que o progresso da resolução puzzle possa ser monitorado. Como no caso do puzzle dos objetos fragmentados, onde enquanto o jogador está com o puzzle em resolução, é exibido uma borda amarela ao redor da tela, e assim que o tempo necessário é atingido e o puzzle é resolvido, uma borda verde é apresentada por alguns segundos. Essas bordas têm por objetivo dar um *feedback* visual ao jogador, indicando que ele está no caminho correto. Por fim, as linhas 35 e 36 são executadas quando o tempo mínimo é atingido, indicado a resolução do puzzle e chamando o método `OnConditionMet`.

Quadro 6 - Classe abstrata OneTimePuzzle

```
5   public abstract class OneTimePuzzle : Puzzle
6   {
7       protected float TimeThreshold = 2f;
8       private bool _isCompleted;
9       private float _timer;
10
            1 usage   1 override   Matheus Navarro Nienow
11      protected abstract void OnIsConditionMet(float timer);
12
            Event function   Matheus Navarro Nienow
13      protected void Update()
14      {
15          if (_isCompleted)
16          {
17              return;
18          }
19
20          var isResolved = IsConditionMet();
21          if (!isResolved)
22          {
23              _timer = 0;
24              OnConditionNotMet();
25              return;
26          }
27
28          _timer += Time.deltaTime;
29          OnIsConditionMet(_timer);
30          if (_timer < TimeThreshold)
31          {
32              return;
33          }
34
35          _isCompleted = true;
36          OnConditionMet();
37      }
38  }
```

Fonte: elaborado pelo autor.

Para validar a resolução do puzzle é comparado a posição e a direção da câmera em relação a posição e a direção do objeto alvo. Apenas quando essas duas condições estão válidas é possível visualizar a ilusão de ótica desejada, conforme demonstrado na Figura 10. A posição é considerada válida quando dois dos eixos da câmera estão dentro de uma faixa de variação em relação aos mesmos eixos do objeto alvo. Essa verificação é feita no método `IsCameraPositionCorrect`, apresentado no Quadro 7. Obtém-se as posições da câmera e do objeto alvo nas linhas 47 e 48 e então calcula-se a diferença dos valores absolutos dos eixos nas linhas 50 e 54. Após isso, nas linhas 56 e 57 são feitas as comparações com o limite de variação desses valores e o resultado é retornado na linha 59.

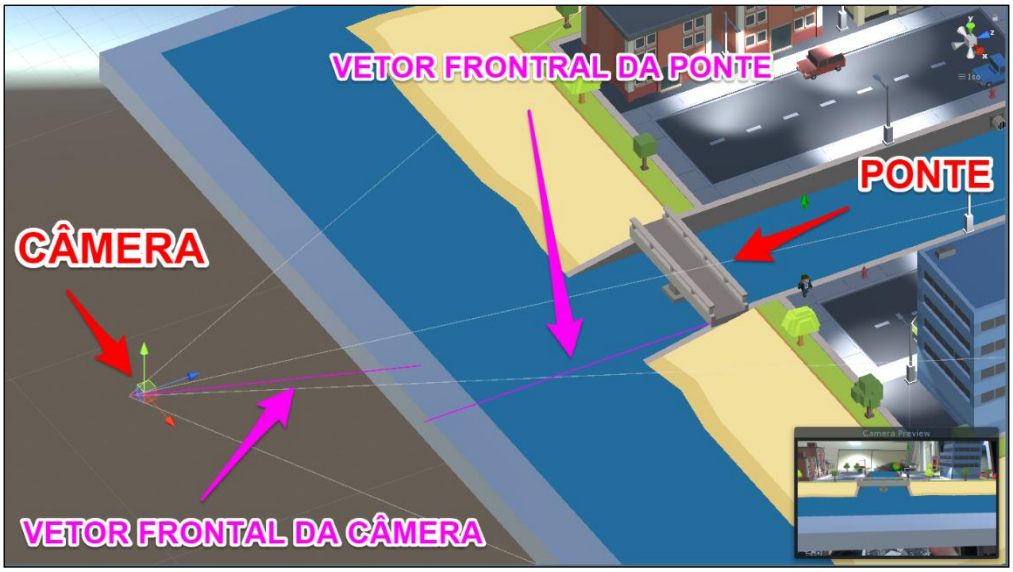
Quadro 7 - Método responsável pela validação da posição da câmera e do objeto alvo

```
44          public static bool IsCameraPositionCorrect(Camera cam, GameObject outOfPathBlock, float cameraXThreshold,
45              float cameraYThreshold, Axis axis)
46          {
47              var camPosition = cam.gameObject.transform.position;
48              var blockPosition = GetPosition(outOfPathBlock);
49
50              var deltaLength = axis == Axis.X
51                  ? AbsDifference(camPosition.z, blockPosition.z)
52                  : AbsDifference(camPosition.x, blockPosition.x);
53
54              var deltaY = AbsDifference(camPosition.y, blockPosition.y);
55
56              var isYAxisCorrect = deltaY < cameraYThreshold;
57              var isLengthAxisCorrect = deltaLength < cameraXThreshold;
58
59              return isYAxisCorrect && isLengthAxisCorrect;
60          }
```

Fonte: elaborado pelo autor.

A validação da direção da câmera em relação ao objeto alvo é feita calculando o produto escalar entre seus vetores frontais, conforme demonstra a Figura 18. O produto escalar de vetores opostos resulta no valor -1 enquanto vetores iguais resulta no valor 1 (UNITY TECHNLOGIES, 2019). Verifica-se então o resultado numa faixa de variação próximo de -1, que é parametrizado e pode ser definida no inspetor do Unity.

Figura 18 - Visualização dos vetores direcionais da câmera e da ponte




Fonte: elaborado pelo autor.

O método `IsCameraDirectionCorrect` é responsável por fazer o cálculo e a verificação desses vetores direcionais. Obtém-se o vetor da câmera e do objeto alvo nas linhas 33 e 34 do método, exibido no Quadro 8. Em seguida é calculado o produto escalar utilizando o método `Vector3.Dot` que faz parte das bibliotecas padrões do Unity. Para fazer esse cálculo é utilizado o valor normalizado dos vetores, também chamado de vetor unitário. Por fim, é utilizado os parâmetros de variação para validar o resultado nas linhas 38, 39 e 41.

Quadro 8 - Método responsável pela verificação da direção da câmera e do objeto alvo

```
30      public static bool IsCameraDirectionCorrect(Camera cam, GameObject outOfPathBlock,
31          Direction outOfPathBlockDirection, float directionThreshold)
32      {
33          var camDirectionVector = cam.transform.forward;
34          var outOfPathBlockDirectionVector = GetDirection(outOfPathBlock, outOfPathBlockDirection);
35
36          var dotResult = Vector3.Dot( lhs: outOfPathBlockDirectionVector.normalized,  rhs: camDirectionVector.normalized) * -1;
37
38          var maxValue = (1 + directionThreshold);
39          var minValue = (1 - directionThreshold);
40
41          return dotResult < maxValue && dotResult > minValue;
42      }
```

Fonte: elaborado pelo autor.

A classe `PuzzleIncompletePath` herda da classe `AlwaysOnPuzzle` e é responsável pelo puzzle do caminho incompleto utilizado na ponte do primeiro nível. Conforme mostra o Quadro 9, quando o puzzle é resolvido ela executa um comando nas linhas 53 e 63 e envia para seus `IObserver` um evento com o *enum* PuzzleStatus `Solved` e o nome do alvo do puzzle nas linhas 54 e 64. O comando executado é o que faz com que o obstáculo do caminho seja desativado para o personagem passar. Esse comando segue o padrão de projeto Command, que tem por objetivo encapsular um comando para que não seja necessário saber os detalhes de sua implementação para ser executado (GAMMA, 1995). Isso permite que o resultado da resolução do puzzle possa ser facilmente configurado.

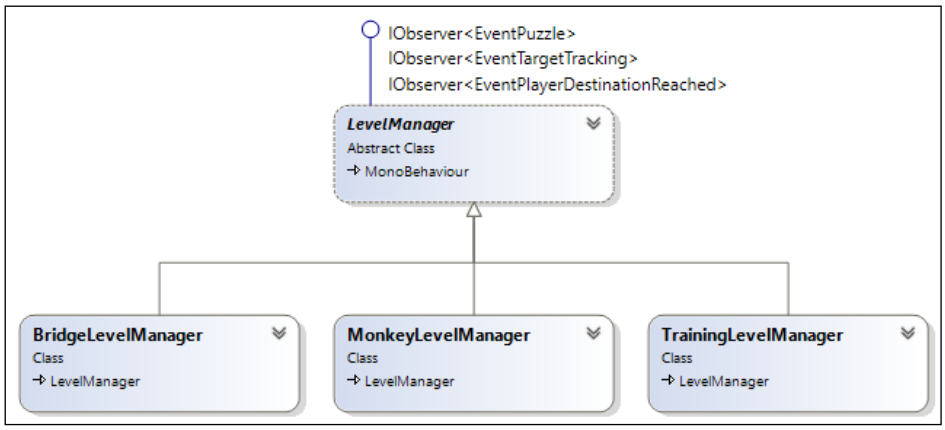
Quadro 9 – Principais métodos da classe `PuzzleIncompletePath`

```
                0+2 usages   Matheus Navarro Nienow
49 ↑0           protected override void OnConditionMet()
50              {
51                  if (!_wasPassageAllowed)
52                  {
53                      _command.Execute();
54                      NotifyOnNext(new EventPuzzle(PuzzleStatus.Solved, outOfPathBlock.name));
55                  }
56                  _wasPassageAllowed = true;
57              }
58
                0+2 usages   Matheus Navarro Nienow
59 ↑0           protected override void OnConditionNotMet()
60              {
61                  if (_wasPassageAllowed)
62                  {
63                      _command.Undo();
64                      NotifyOnNext(new EventPuzzle(PuzzleStatus.NotSolved, outOfPathBlock.name));
65                  }
66                  _wasPassageAllowed = false;
67              }
```

Fonte: elaborado pelo autor.

O gerenciamento dos níveis é feito por classes que herdam da classe abstrata `LevelManager`. Essas classes são responsáveis por gerenciar as mensagens que são apresentadas ao jogador, assim como os objetivos do personagem no nível. A classe `LevelManager` define o comportamento comum entre elas, assim como exige que seus filhos implementem certos métodos abstratos. A Figura 19 exibe o diagrama dessas classes.

Figura 19 - Diagrama de classes da estrutura de LevelManager




Fonte: elaborado pelo autor.

O destino do personagem é controlado por um sistema de *checkpoints*. Existem `GameObjects` específicos da cena que possuem seu nome definido como “CheckPoint” concatenado ao seu índice, conforme mostra a Figura 20. Esses objetos representam o caminho que o personagem seguirá no decorrer do nível.

Figura 20 - *Checkpoints* do segundo nível


Fonte: elaborado pelo autor.

O `LevelManager` utiliza esses objetos para definir os destinos do personagem na cena. O Quadro 10 exibe o método `SetNextCheckPoint`, responsável por buscar o próximo *checkpoint* e defini-lo como destino do personagem. O método busca os *checkpoints* pelo seu nome na linha 66, atribui esse *checkpoint* como o destino do personagem na linha 71 e então move o personagem até ele na linha 71. A variável `_checkPointIndex` é utilizada para controlar qual *checkpoint* é o destino atual do personagem. Esse método possui um *overload*, fazendo com que tenha duas assinaturas, uma sem parâmetros onde o método apenas busca o próximo *checkpoint* incrementando a variável `_checkPointIndex`, e outra assinatura onde o índice é parametrizado e esse valor então é atribuído à `_checkPointIndex`.

Quadro 10 - Métodos para buscar os *checkpoints*

```
53      protected bool SetNextCheckPoint()
54      {
55          var nextIndex = _checkPointIndex + 1;
56          var result = SetNextCheckPoint(nextIndex);
57          if (result)
58          {
59              _checkPointIndex = nextIndex;
60          }
61          return result;
62      }
63
                Frequently called  5 usages  Matheus Navarro Nienow
64      protected bool SetNextCheckPoint(int index)
65      {
66          var destination = GameObject.Find("CheckPoint" + index);
67          if (destination == null)
68          {
69              return false;
70          }
71          playerController.Destination = destination;
72          playerController.Move();
73          _checkPointIndex = index;
74          return true;
75      }
```

Fonte: elaborado pelo autor.

O `LevelManager` é um `IObserver` do tipo `EventPlayerDestinationReached`, uma classe do tipo evento que o `NavMeshAgentController` envia quando o personagem chega ao seu destino. Seu código está exibido no Quadro 11. Ao receber o evento o `LevelManager` executa o método `SetNextPoint` na linha 136, para manter o personagem em movimento rumo ao seu próximo *checkpoint*, criando um loop de movimentação do personagem. Quando um novo *checkpoint* não é encontrado, significa que esse é o fim do jogo. Então na linha 139 o método `EndGame` é executado, exibindo o painel final do nível.

Quadro 11 - Método receptor do evento `EventPlayerDestinationReached`

```
122     public void OnNext(EventPlayerDestinationReached destinationReachedEvent)
123     {
124         if (destinationReachedEvent?.Destination != null)
125         {
126             var destinationName = destinationReachedEvent.Destination != null
127                 ? destinationReachedEvent.Destination.name
128                 : null;
129
130             var isSpecialCheckPoint = HandleSpecialCheckPoint(destinationName);
131             if (isSpecialCheckPoint)
132             {
133                 return;
134             }
135         }
136         var result = SetNextCheckPoint();
137         if (!result)
138         {
139             EndGame();
140         }
141     }
```

Fonte: elaborado pelo autor.

O método abstrato `HandleSpecialCheckPoint` permite que *checkpoints* específicos sejam identificados. Nesses casos, o loop de movimentação é quebrado nas linhas 130 a 134, para que algum método específico seja executado, ficando de responsabilidade dos gerenciadores dos níveis decidirem o que deve ser executado. Por exemplo, o Quadro 12 exibe a implementação feita pela classe `MonkeyLevelManager`. Nessa implementação, ao personagem chegar no quinto

*checkpoint*, na linha 152 é iniciado uma corrotina para executar o script responsável pela parte do nível onde o jogador deve encontrar os objetos de metais.

Quadro 12 - Implementação do método `HandleSpecialCheckPoint` da classe `MonkeyLevelManager`

```
147    protected override bool HandleSpecialCheckPoint(string destinationName)
148    {
149        switch (destinationName)
150        {
151            case "CheckPoint5":
152                StartCoroutine( routine: MetalGarbageScript());
153                return true;
154            case "CheckPoint9":
155                StartCoroutine( routine: PlasticGarbageScript());
156                return true;
157            default:
158                return false;
159        }
160    }
```

Fonte: elaborado pelo autor.

O método `MetalGarbageScript` é um exemplo de como o jogo gerencia a execução dos níveis. No Quadro 13 é possível visualizar sua implementação. As mensagens são enviadas ao painel da interface nas linhas 88, 92 e 95 e então uma determinada quantidade de segundos é esperada nas linhas 90 e 93, antes de definir as próximas mensagens. Entre essas mensagens outros métodos podem ser executados, como nas linhas 98 e 99.

Quadro 13 - Método `MetalGarbageScript`

```
86     private IEnumerator MetalGarbageScript()
87     {
88         SetHelperMessage("O personagem chegou na lixeira AMARELA. Que tipos de produtos devem ser jogados " +
89                          "nessa leixeira?");
90         yield return new WaitForSeconds(5);
91
92         SetHelperMessage("Existem diversos produtos espalhados ao redor do mapa.");
93         yield return new WaitForSeconds(5);
94
95         SetHelperMessage("Para poder avançar, visualize dois produtos feitos do material que devem ser jogados " +
96                          "nessa lixeira. O painel superior direito indica a quantidade de produtos encontrados.");
97
98         ActivateScorePanel();
99         ActivateMetalPuzzle();
100    }
```

Fonte: elaborado pelo autor.

## 4 RESULTADOS

Este capítulo apresenta os dois tipos de testes realizados com o jogo. Na primeira seção são discutidos os testes de funcionalidades do jogo, realizados durante o desenvolvimento. Na última seção são apresentados os resultados dos testes realizados com usuários.

### 4.1 TESTES DE FUNCIONALIDADES

Durante o desenvolvimento do jogo foram realizados testes constantes para garantir que as funcionalidades estivessem de acordo com o esperado. Foram realizados testes tanto no emulador do Unity quanto num dispositivo real. O dispositivo móvel utilizado para o desenvolvimento do jogo foi um Samsung Galaxy S7 Edge, que é do tipo *handheld display*. Os testes levaram em considerações a forma como os usuários poderiam utilizar o jogo. Foram observados diversos aspectos, como tamanho do mapa, altura do mapa em relação ao marcador, tamanho e tempo das mensagens apresentadas aos jogadores, entre outros aspectos.

Nessa etapa foi identificado que o Vuforia não permite a visão ortográfica dos objetos 3D por conta da interação com o mundo real. Isso fez com que a ideia inicial de utilizar objetos impossíveis como principal ilusão no jogo fosse descartada, visto que as ilusões não obtiveram um resultado agradável numa câmera com visão em perspectiva. Por conta disso optou-se por utilizar ilusões simples e intuitivas, como a visualização de produtos fragmentados, que assim como os objetos impossíveis, é uma ilusão do tipo paradoxo e de classe cognitiva.

O desenvolvimento do jogo foi realizado de forma iterativa, com constante refatoração de código para atender a novos requisitos do jogo. As funcionalidades foram construídas para atender a objetivos específicos e foram refatoradas

para atender objetivos de forma genérica conforme a necessidade. Assim, testes foram realizados constantemente para garantir que as funcionalidades continuavam de acordo em todas as partes do jogo.

## 4.2 TESTES DE UTILIZAÇÃO PRELIMINAR

Após finalizado o desenvolvimento foram realizados testes preliminares com 5 usuários. A aplicação dos testes não teve um local específico, 3 dos testes foram realizados na presença do autor utilizando o dispositivo de desenvolvimento, enquanto 2 foram realizados nos dispositivos dos usuários sem a presença do autor. Nesses casos os usuários estavam acompanhados de outro usuário que havia testado o jogo previamente com o autor e era capaz de instruí-los na realização dos testes. Primeiramente foi apresentado o jogo e seu objetivo e em seguida foi aplicado um questionário divido em 3 etapas. O questionário completo está exibido no Apêndice B.

A primeira etapa do questionário consistia em perguntas para avaliar o perfil do usuário. Pôde-se observar que 80% dos usuários possuem entre 20 e 24 anos, sendo 60% do sexo masculino. Destes, 40% possuem ensino superior completo, 40% possuem ensino superior incompleto e 20% possuem ensino médio completo. Todos os usuários utilizam dispositivos móveis frequentemente e 60% deles nunca haviam utilizado alguma aplicação com RA. O Quadro 14 exibe as respostas obtidas.

Quadro 14 - Perfil dos usuários

| **Pergunta** | **Respostas** | **Quantidade** |
|---|---|---|
| Faixa etária | De 20 a 24 anos<br>De 40 a 49 anos | 80%<br>20% |
| Sexo | Feminino<br>Masculino | 40%<br>60% |
| Qual é o seu grau de escolaridade? | Ensino médio completo – 2º grau<br>Ensino superior incompleto<br>Ensino superior completo | 20%<br>40%<br>40% |
| Você utiliza dispositivos móveis com qual frequência? | Frequentemente | 100% |
| Já utilizou aplicações de Realidade Aumentada (RA)? | Sim<br>Não | 40%<br>60% |

Fonte: elaborado pelo autor.

A segunda etapa do questionário consistia em um passo-a-passo para o usuário seguir e realizar as tarefas do jogo. Essa etapa tinha como objetivos instruir o jogador a iniciar o jogo, reconhecer o marcado utilizando a câmera do dispositivo e jogar todos os três níveis disponíveis. Como esperado, todas as respostas foram “Concluído”, visto que todas as instruções eram simples. Por fim, a etapa final apresentava perguntas de cunho avaliativo sobre dificuldade, jogabilidade, facilidade de interação e eficiência. As respostas foram registradas na escala Likert. O Quadro 15 mostra os resultados obtidos nessa etapa.

Quadro 15 - Avaliação dos usuários

| **Pergunta** | **Respostas (Likert)** | **Quantidade** |
|---|---|---|
| Você conseguiu concluir as tarefas dessa pesquisa com facilidade?<br>(menor é melhor) | 2<br>3 | 40%<br>60% |
| Como você classifica a interação com o jogo através da câmera?<br>(maior é melhor) | 3<br>4<br>5 | 40%<br>20%<br>40% |
| Como você classifica a jogabilidade geral do jogo?<br>(maior é melhor) | 3<br>4<br>5 | 20%<br>60%<br>20% |
| Como você classifica a inserção do tema "Reciclagem" no jogo?<br>(maior é melhor) | 4<br>5 | 20%<br>80% |
| Você acha que o CidadaniaAR cumpriu seu objetivo de desenvolver um jogo de puzzle com realidade aumentada e ilusão de ótica?<br>(maior é melhor) | 5 | 100% |

Na última pergunta foi aberto um espaço para os usuários registrarem algum comentário geral, crítica ou sugestão sobre o jogo. Foram obtidas duas respostas para essa questão. Uma delas foi feita por um usuário do sexo feminino, na faixa etária de 40 a 49 anos, com ensino superior completo, que indicou que como professora achou o jogo muito criativo e um ótimo material para fixação da matéria para uma criança. A segunda resposta foi feita por um usuário do sexo masculino, na faixa etária de 20 a 24 anos, com ensino superior incompleto, sugerindo alguns pontos de melhoria para o jogo, como aumento da velocidade do personagem, permitir um maior controle da função de zoom para visualizar mais facilmente o mapa, assim como o aumento do tamanho do diamante que indica a posição do personagem, pois ele sentiu um pouco de dificuldade de encontrar o personagem em algumas partes do mapa.

Os resultados são satisfatórios, todos os usuários concordaram que o jogo alcançou seus objetivos. Alguns usuários tiveram mais dificuldades que outros para concluir algumas das tarefas do jogo, o que indica que há pontos a serem melhorados no jogo. As sugestões de melhorias podem ser utilizadas para dar continuidade ao trabalho.

## 5 CONCLUSÕES

Com base nos resultados obtidos, o jogo alcançou seu objetivo de utilizar a RA para criar mecânicas de jogabilidades baseadas na ilusão de ótica. O objetivo específico de criar um sistema de interação com o jogo utilizando elementos do mundo real foi alcançado com a necessidade de movimentação do dispositivo e/ou do marcador para completar as tarefas do jogo. A incorporação do tema de reciclagem no jogo se mostrou satisfatória, incluindo uma avaliação positiva de uma professora indicando que o jogo pode ser uma ótima ferramenta de fixação de matéria para crianças. As funcionalidades do jogo funcionaram de acordo com o esperado.

O Unity se mostrou uma ótima ferramenta e atendeu a todos as necessidades do desenvolvimento do jogo. A utilização da linguagem C# fez com que a curva de aprendizagem do Unity fosse baixa, enquanto a grande comunidade do Unity facilitou a solução de problemas durante o desenvolvimento. O Vuforia é uma ótima biblioteca para a criação de aplicações em RA, sendo bastante simples, intuitiva e eficiente. O Blender foi capaz de atender as necessidades de customização do jogo em relação aos modelos 3D utilizados.

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

a) incrementar a quantidade de níveis;
b) adicionar puzzles que utilizam outras ilusões de ótica;
c) adicionar puzzles que utilizem marcadores adicionais para sua resolução;
d) adicionar efeitos sonoros e músicas;
e) aplicar e avaliar o desempenho do jogo no auxílio do tema em turmas do ensino fundamental;
f) incorporar outros temas, como conscientização no trânsito.

## REFERÊNCIAS


APPLE. **Realidade aumentada:** para iOS. 2019. Disponível em: <https://www.apple.com/br/ios/augmented-reality/>. Acesso em: 26 maio 2019.

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

AZUMA, Ronald et al. Recent advances in augmented reality. **IEEE computer graphics and applications**, v. 21, n. 6, p. 34-47, 2001.

BORISENKO, Aleksandr Ivanovich et al. **Vector and tensor analysis with applications**. Courier Corporation, 1968.

BROMSKLOSS. **File:Penrose triangle.svg**. Disponível em: <https://en.wikipedia.org/wiki/File:Penrose_triangle.svg>. Acesso em: 07 abr. 2019.

CARMIGNIANI, Julie et al. Augmented reality technologies, systems and applications. **Multimedia tools and applications**, v. 51, n. 1, p. 341-377, 2011.

CLAIR, Bryan. **File:Necker-cube.svg.** Disponível em: <https://mathstat.slu.edu/escher/index.php?title=File:Necker-cube.svg&limit=50>. Acesso em: 21 maio 2019.

IBISWORLD. **Global Movie Production & Distribution Industry**. 2018. Disponível em: <https://www.ibisworld.com/industry-trends/global-industry-reports/other-community-social-personal-service-activities/movie-production-distribution.html>. Acesso em: 06 abr. 2019.

IFPI. **Global Music Report 2019:** State of the Industry. Zurich: Ifpi, 2019. 40 p.

GAMMA, Erich. **Design patterns: elements of reusable object-oriented software**. Pearson Education India, 1995.

GREGORY, Richard L. Putting Illusions in their Place. **Perception**, [s.l.], v. 20, n. 1, p.1-4, fev. 1991. SAGE Publications.

LEITÃO, Rui Manuel Vieira. **APRENDIZAGEM BASEADA EM JOGOS: REALIDADE AUMENTADA NO ENSINO DE SÓLIDOS GEOMÉTRICOS.** 2013. 77 f. Dissertação (Mestrado) - Curso de Mestrado em Expressão Gráfica e Audiovisual, Universidade Aberta, Lisboa, 2013.

MARNER, Michael R. et al. Spatial user interfaces for large-scale projector-based augmented reality. **IEEE computer graphics and applications**, v. 34, n. 6, p. 74-82, 2014.

MICROSOFT. **HoloLens 2:** A new vision for computing. Disponível em: <https://www.microsoft.com/en-us/hololens/hardware>. Acesso em: 26 maio 2019.

NEWZOO (Holanda). **2018 Global Games Market Report (FREE VERSION)**: Trends, insights, and projections toward 2021. Amsterdã: Newzoo, 2019. 25 p.

PASHLEY, Peter. **Ustwo at Nordic Game 2014:** Making of Monument Valley in Unity. 2014. (47m20s). Disponível em: <https://www.youtube.com/watch?v=mCCC9hQm6MM&t=1146s>. Acesso em: 29 mar. 2019.

SILVA, Manoela et al. **AR Jigsaw Puzzle:** Potencialidades de Uso da Realidade Aumentada no Ensino de Geografia. Recife: Universidade Federal de Pernambuco, 2014. 10 p.

SPIEGEL, Murray R.; LIPSCHUTZ, Seymour; LIU, John. **Mathematical Handbook of Formulas and Tables.** 3. ed. [s. L.]: Mcgraw-hill, 2009. 289 p.

TACH, Dave. **Monument Valley and more games win Apple Design Awards**. 2014. Disponível em: <https://www.polygon.com/2014/6/3/5776986/games-apple-design-awards-2014>. Acesso em: 30 mar. 2019.

UNITY TECHNOLOGIES. **Vector3.Dot.** 2019. Disponível em: <https://docs.unity3d.com/ScriptReference/Vector3.Dot.html>. Acesso em: 06 out. 2019.

UNITY (Estados Unidos da América). **Public Relations:** Unity Growth Facts. 2019. Disponível em: <https://Unity.com/public-relations>. Acesso em: 07 abr. 2019.

USTWO GAMES (London). **Monument Valley**. Disponível em: <https://www.monumentvalleygame.com/mv1>. Acesso em: 30 mar. 2019.

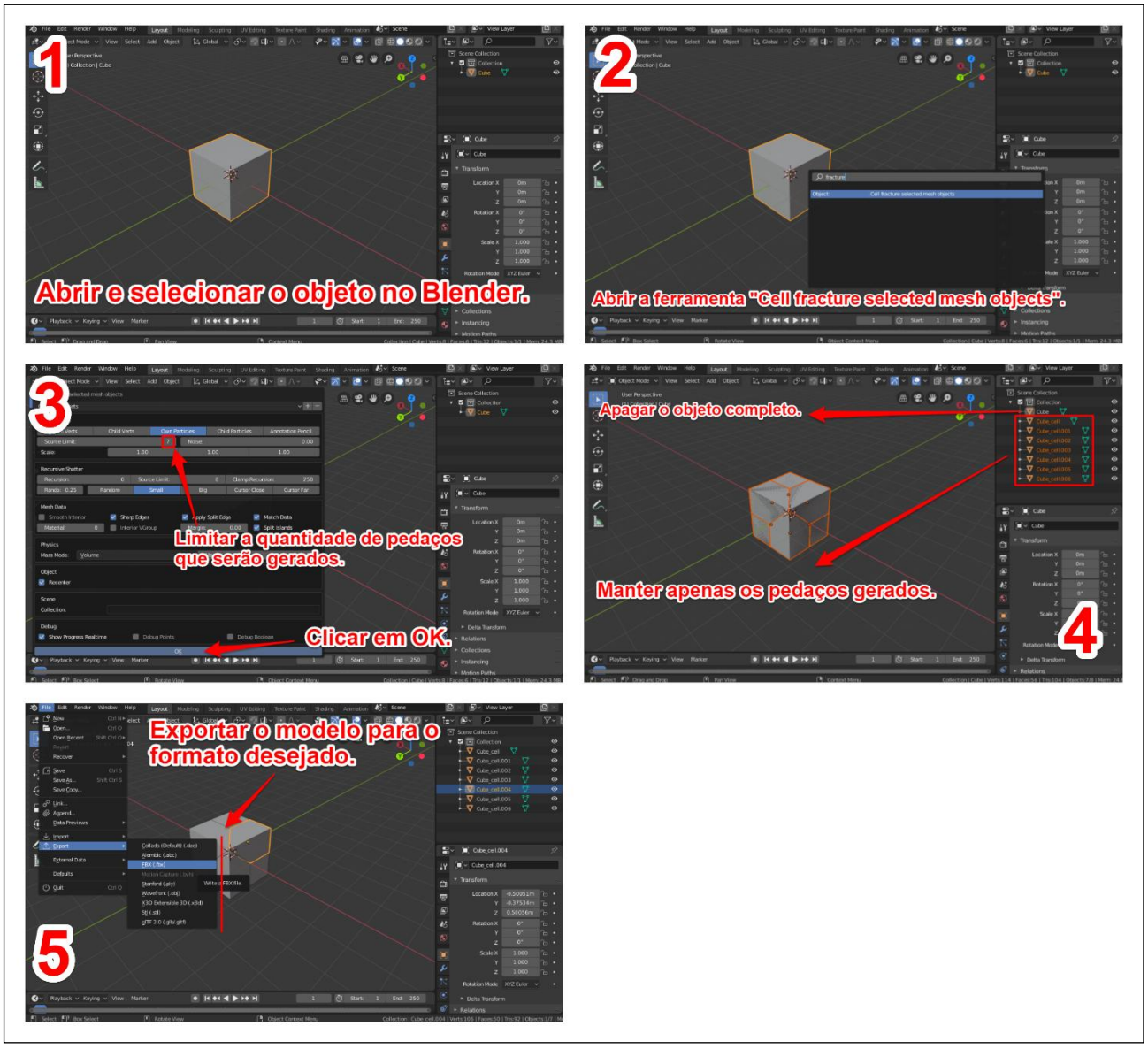
VITICCI, Federico. **The 2014 Apple Design Award Winners**. 2014. Disponível em: <https://www.macstories.net/news/the-2014-apple-design-award-winners/>. Acesso em: 07 abr. 2019.

ZHOU, Feng; DUH, Henry Been-Lirn; BILLINGHURST, Mark. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In: **Proceedings of the 7th IEEE/ACM international symposium on mixed and augmented reality**. IEEE Computer Society, 2008. p. 193-202.

## APÊNDICE A – PASSO A PASSO DA FRAGMENTAÇÃO DE OBJETOS NO BLENDER

Neste apêndice é apresentado o processo de fragmentação de modelos utilizando a ferramenta de modelagem Blender. A ferramenta possui uma funcionalidade para separadas um objeto em diversos fragmentos. A Figura 21 exibe o passo a passo para executar esse processo. Após isso, o modelo é importado no Unity e seus pedaços podem ser movimentados livremente para formar a ilusão de ótica. O ideal é mover todas as peças no mesmo eixo, de maneira que para visualizar o objeto como um todo a observação deva ser feita a partir desse eixo.

Figura 21 - Passo a passo da fragmentação de modelos no Blender

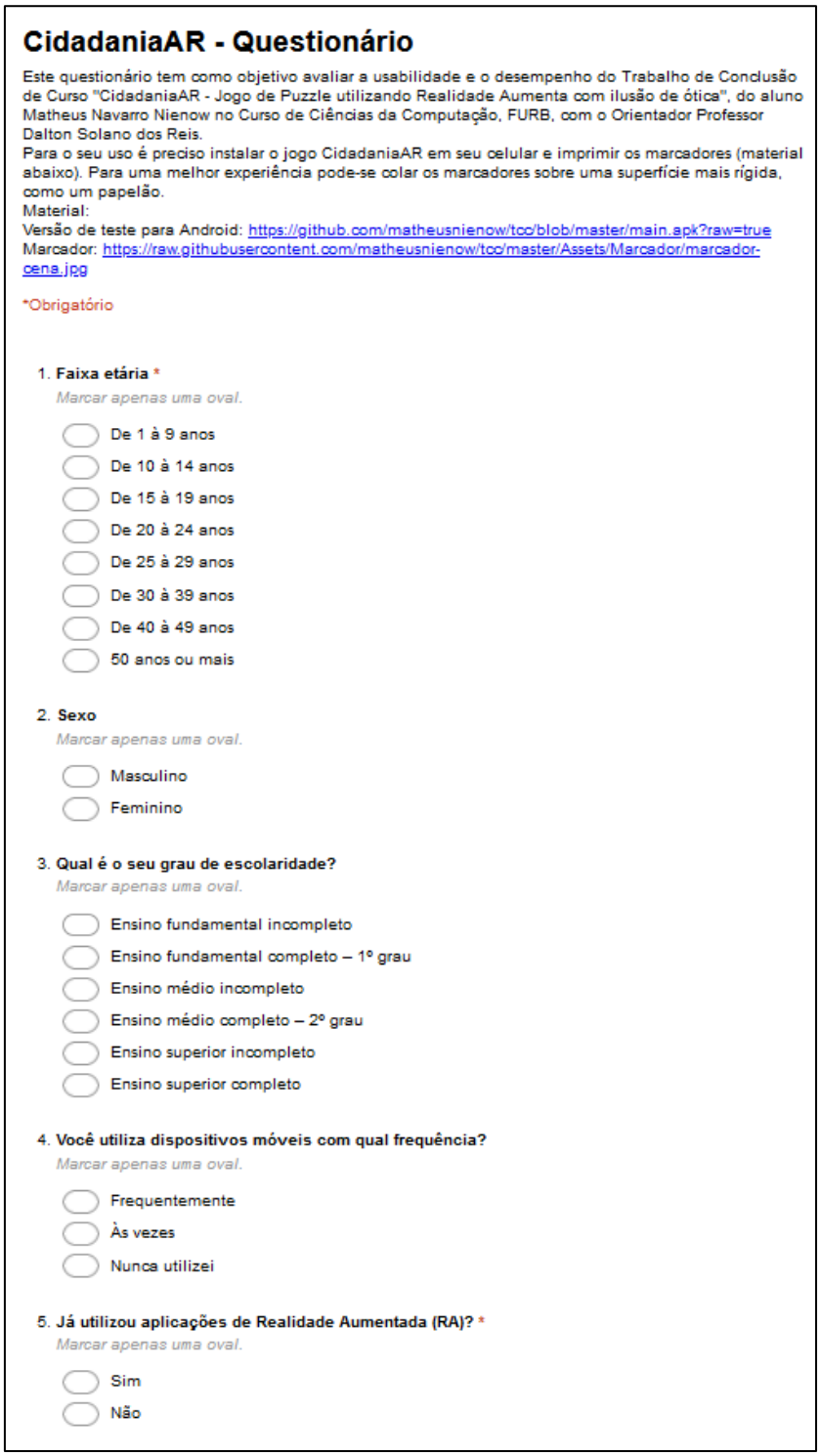



Fonte: elaborado pelo autor.

## APÊNDICE B – QUESTIONÁRIO DE AVALIAÇÃO

Neste apêndice é apresentado o questionário de avaliação utilizado na realização dos testes preliminares do jogo. O questionário é dividido em três etapas. A primeira etapa, exibida na Figura 22, apresenta questionamentos a respeito do perfil do usuário. A segunda etapa, exibida na Figura 23, dá instruções quanto a utilização do jogo. A terceira e última etapa, exibida na Figura 24, tem por objetivo fazer a avaliação do jogo.

Figura 22 - Primeira etapa do questionário



**CidadaniaAR - Questionário**

Este questionário tem como objetivo avaliar a usabilidade e o desempenho do Trabalho de Conclusão de Curso "CidadaniaAR - Jogo de Puzzle utilizando Realidade Aumenta com ilusão de ótica", do aluno Matheus Navarro Nienow no Curso de Ciências da Computação, FURB, com o Orientador Professor Dalton Solano dos Reis.
Para o seu uso é preciso instalar o jogo CidadaniaAR em seu celular e imprimir os marcadores (material abaixo). Para uma melhor experiência pode-se colar os marcadores sobre uma superfície mais rígida, como um papelão.
Material:
Versão de teste para Android: https://github.com/matheusnienow/tcc/blob/master/main.apk?raw=true
Marcador: https://raw.githubusercontent.com/matheusnienow/tcc/master/Assets/Marcador/marcador-cena.jpg

*Obrigatório

1. **Faixa etária** *
   *Marcar apenas uma oval.*
   - ◯ De 1 à 9 anos
   - ◯ De 10 à 14 anos
   - ◯ De 15 à 19 anos
   - ◯ De 20 à 24 anos
   - ◯ De 25 à 29 anos
   - ◯ De 30 à 39 anos
   - ◯ De 40 à 49 anos
   - ◯ 50 anos ou mais

2. **Sexo**
   *Marcar apenas uma oval.*
   - ◯ Masculino
   - ◯ Feminino

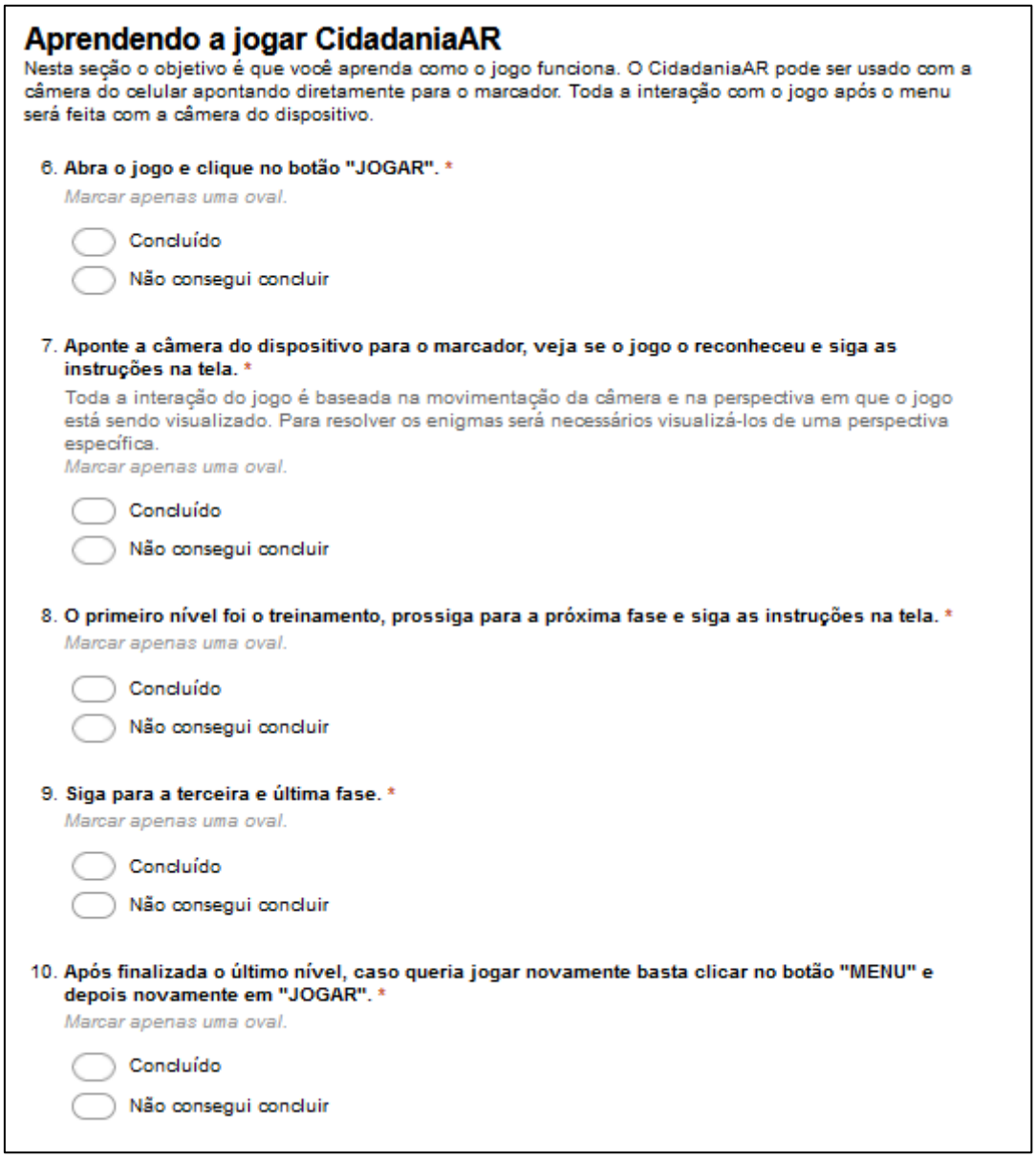
3. **Qual é o seu grau de escolaridade?**
   *Marcar apenas uma oval.*
   - ◯ Ensino fundamental incompleto
   - ◯ Ensino fundamental completo – 1º grau
   - ◯ Ensino médio incompleto
   - ◯ Ensino médio completo – 2º grau
   - ◯ Ensino superior incompleto
   - ◯ Ensino superior completo

4. **Você utiliza dispositivos móveis com qual frequência?**
   *Marcar apenas uma oval.*
   - ◯ Frequentemente
   - ◯ Às vezes
   - ◯ Nunca utilizei

5. **Já utilizou aplicações de Realidade Aumentada (RA)?** *
   *Marcar apenas uma oval.*
   - ◯ Sim
   - ◯ Não

Fonte: elaborado pelo autor.

Figura 23 - Segunda etapa do questionário



## Aprendendo a jogar CidadaniaAR

Nesta seção o objetivo é que você aprenda como o jogo funciona. O CidadaniaAR pode ser usado com a câmera do celular apontando diretamente para o marcador. Toda a interação com o jogo após o menu será feita com a câmera do dispositivo.

6. **Abra o jogo e clique no botão "JOGAR".** *
   *Marcar apenas uma oval.*
   - ◯ Concluído
   - ◯ Não consegui concluir

7. **Aponte a câmera do dispositivo para o marcador, veja se o jogo o reconheceu e siga as instruções na tela.** *
   Toda a interação do jogo é baseada na movimentação da câmera e na perspectiva em que o jogo está sendo visualizado. Para resolver os enigmas será necessários visualizá-los de uma perspectiva específica.
   *Marcar apenas uma oval.*
   - ◯ Concluído
   - ◯ Não consegui concluir

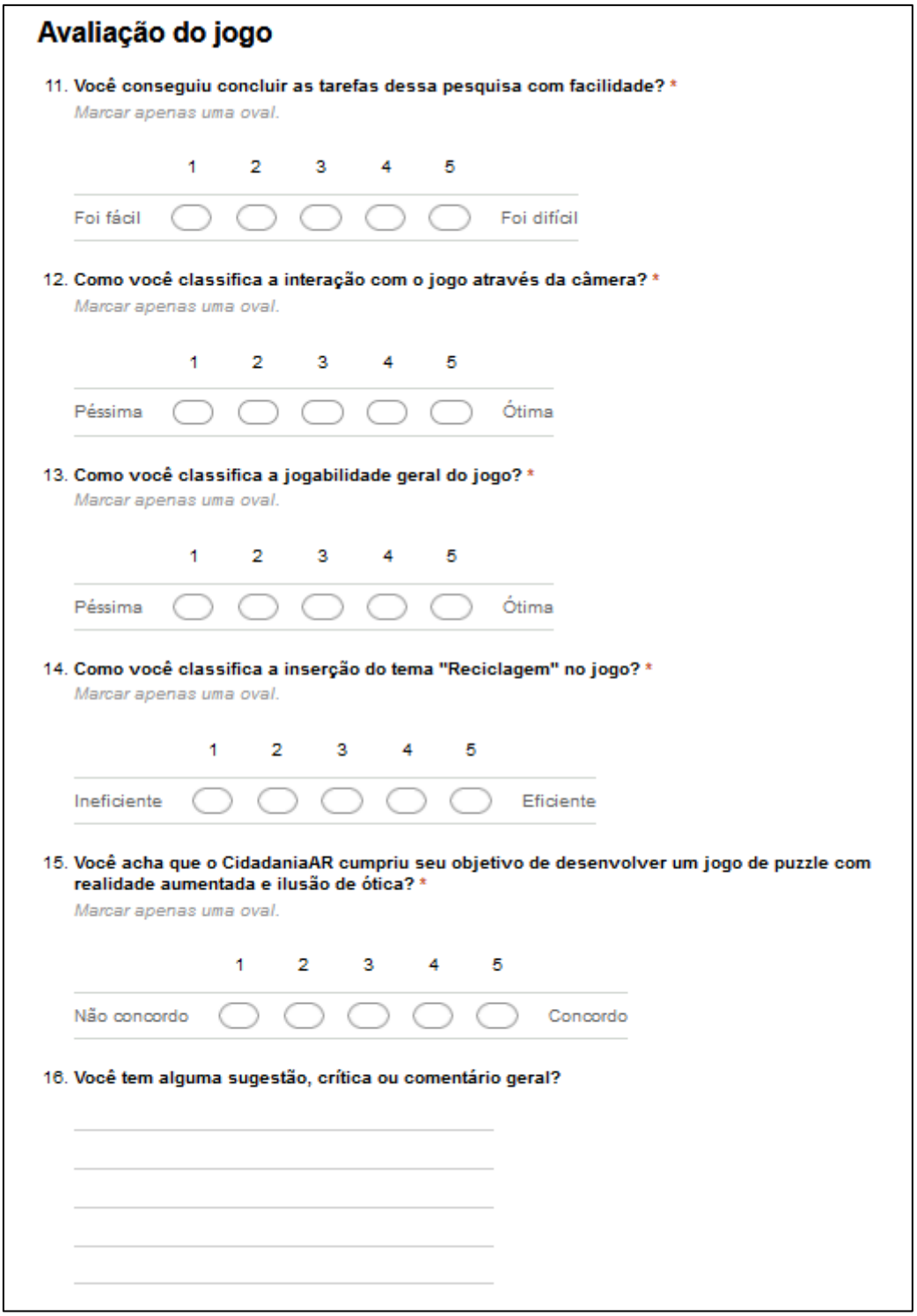
8. **O primeiro nível foi o treinamento, prossiga para a próxima fase e siga as instruções na tela.** *
   *Marcar apenas uma oval.*
   - ◯ Concluído
   - ◯ Não consegui concluir

9. **Siga para a terceira e última fase.** *
   *Marcar apenas uma oval.*
   - ◯ Concluído
   - ◯ Não consegui concluir

10. **Após finalizada o último nível, caso queria jogar novamente basta clicar no botão "MENU" e depois novamente em "JOGAR".** *
    *Marcar apenas uma oval.*
    - ◯ Concluído
    - ◯ Não consegui concluir

Fonte: elaborado pelo autor.

Figura 24 - Etapa final do questionário



## Avaliação do jogo

11. **Você conseguiu concluir as tarefas dessa pesquisa com facilidade?** *
*Marcar apenas uma oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Foi fácil | ◯ | ◯ | ◯ | ◯ | ◯ | Foi difícil |

12. **Como você classifica a interação com o jogo através da câmera?** *
*Marcar apenas uma oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Péssima | ◯ | ◯ | ◯ | ◯ | ◯ | Ótima |

13. **Como você classifica a jogabilidade geral do jogo?** *
*Marcar apenas uma oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Péssima | ◯ | ◯ | ◯ | ◯ | ◯ | Ótima |

14. **Como você classifica a inserção do tema "Reciclagem" no jogo?** *
*Marcar apenas uma oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Ineficiente | ◯ | ◯ | ◯ | ◯ | ◯ | Eficiente |

15. **Você acha que o CidadaniaAR cumpriu seu objetivo de desenvolver um jogo de puzzle com realidade aumentada e ilusão de ótica?** *
*Marcar apenas uma oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Não concordo | ◯ | ◯ | ◯ | ◯ | ◯ | Concordo |

16. **Você tem alguma sugestão, crítica ou comentário geral?**

Fonte: elaborado pelo autor.