UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

JOGO DE SINUCA VIRTUAL COM REALIDADE AUMENTADA

PEDRO HENRIQUE SCHMITT

BLUMENAU
2017

PEDRO HENRIQUE SCHMITT

JOGO DE SINUCA VIRTUAL COM REALIDADE AUMENTADA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2017

JOGO DE SINUCA VIRTUAL COM REALIDADE AUMENTADA

Por

PEDRO HENRIQUE SCHMITT

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof(a). Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Prof(a). Daniel Theisges dos Santos, Mestre – FURB

Blumenau, 05 de julho de 2017

Dedico este trabalho aos meus pais, meus irmãos, a todos os meus amigos, em especial aos que me ajudaram, e ao meu orientador pelo apoio para realização do mesmo.

AGRADECIMENTOS

Aos meus pais Osmar e Eliana aos meus irmãos Maria Clara e Francisco José pela paciência, incentivo e apoio para o desenvolvimento deste trabalho.

Aos meus amigos que contribuíram e deram muito apoio para o desenvolvimento deste trabalho.

À todos os participantes da 6ª Caminhada Franciscana da Juventude em Rodeio, pela energia, força, paz e alegria recebido ao participar do evento durante o desenvolvimento deste trabalho.

Ao professor Aurélio Faustino Hoppe pela ideia do trabalho e por seus comentários durante o desenvolvimento.

Ao meu orientador Dalton Solano dos Reis, por acreditar na realização deste trabalho e por seu apoio e tempo investido no mesmo.

Um dia aprendi que sonhos existem para tornarem-se realidade. E, desde aquele dia, já não durmo pra descansar. Simplesmente durmo para sonhar.

Walt Disney

RESUMO

Este trabalho apresenta o desenvolvimento de um jogo de sinuca com Realidade Aumentada e com a interação do taco físico de sinuca. O jogo foi implementado no *framework* Unity 3D e com a biblioteca Vuforia para a Realidade Aumentada. Foram realizados vários testes e implementações para identificar o movimento do taco físico de sinuca no ambiente real e a interação com a ambiente virtual. Foram realizadas pesquisas e testes para a implementação de uma interface tangível utilizando o taco de sinuca. Além disso, são apresentados testes de marcadores para utilização da Realidade Aumentada, o desenvolvimento de dois simulacros de sinuca e a avaliação do jogo por convidados. Como resultado, foi desenvolvido um jogo de sinuca que pode ser praticado por duas pessoas com interface tangível, que identifica os movimentos realizados pelo taco de sinuca e interagi com o jogo virtual que possui as regras da sinuca.

Palavras-chave: Jogo de sinuca. Realidade aumentada. Unity 3D. Vuforia. Interface tangível.

ABSTRACT

This work presents the development of a pool game with Augmented Reality and an interaction of the physical pool snooker. The game was implemented without structure 3D Unit with a Vuforia library for Augmented Reality. Several tests and implementations were carried out to identify the movement of the physical pool snorkel in the real environment and an interaction with the virtual environment. We conducted research and tests for an implementation of a tangible interface using pool cues. In addition, markers tests for the use of Augmented Reality, the development of two snooker simulations and a guest game evaluation were published. As a result, a pool game has been developed that can be practiced by two people with a tangible interface, which identifies the movements performed by the pool and interacted with the virtual game that has as rules of the pool.

Keywords: Pool game. Increased reality. 3D drive. Vuforia. Tangible interface.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CBBS – Confederação Brasileira de Bilhar e Sinuca

CM – CentíMetro

HMD – Head Mounted Display

JPG – Joint Photographic Group

LED – Light Emitting Diode

RA – Realidade Aumentada

RF – Requisito Funcional

RNF – Requisito Não Funcional

SDK – Software Development Kit

SUMÁRIO

1 INTRODUÇÃO

Desde 2006, com o lançamento do *videogame* Wii, os jogos virtuais, entraram em um novo estágio, levando o jogador á uma nova interação com os jogos virtuais. As empresas da área de jogos sempre tiveram como objetivo fazer jogos mais reais. Para isso, apostam na parte gráfica dos jogos, com imagens, movimentos e sons cada vez mais realistas, passando a sensação ao usuário de ser algo real (FERREIRA, 2008).

Com o console Wii, a forma de deixar os jogos mais reais foi diferente. A empresa Nintendo, fabricante do Wii, apostou na interação do jogador e não na parte gráfica dos jogos (FERREIRA, 2008). Neste videogame os usuários não ficam sentados com um *jostick*, ou até mesmo com um mouse e teclado nas mãos, e passam a interagir com o jogo fazendo movimentos corporais.

Logo após o lançamento, muitos perceberam os benefícios que o Wii poderia oferecer. Jogar tênis, boliche, *baseball*, entre outros esportes, realizando os mesmos movimentos que um jogo real, despertou curiosidade até mesmo em pessoas que não jogavam videogames. Verifica-se alguns benefícios na pesquisa realizada por Finco e Fraga (2012), que analisou os comentários dos usuários do jogo Wii Fit, um jogo do videogame Wii. Muitos usuários comentaram sobre, perda de peso, melhor qualidade de vida e saúde com a utilização do jogo. Finco e Fraga (2012, p. 8) ainda conclui que o jogo "[..]se transformou em uma ferramenta com significativo potencial para conscientização de seus usuários em relação à prática regular de exercício físico e a cuidados com a saúde.". Em uma pesquisa na Inglaterra, o console Wii é o equipamento que deixa os usuários mais felizes, por unir os amigos na sala de casa para se divertir com atividades corporais (FOLHA ONLINE, 2009).

Outra maneira de obter a imersão tecnológica com o mundo virtual é com a Realidade Aumentada (RA), onde são gerados objetos virtuais em um ambiente real. Conforme Faust et al. (2011, p. 2) a RA é implementada de forma que "[..] o cenário real e os objetos virtuais permanecem ajustados, mesmo com a movimentação do usuário no ambiente real". Essa característica da RA tem o objetivo passar a sensação ao usuário, dele estar no mundo virtual de maneira amigável.

A utilização dessa interação do usuário com o mundo virtual, também faz com que o usuário tenha acesso ao um esporte, que no mundo real se tornaria caro e perigoso. Pode-se citar como exemplo o *box*, que praticado no videogame não traz perigo aos usuários, pois o ambiente é virtual, mas os movimentos são os mesmos do esporte real. Outro esporte que tem custo alto com equipamentos e possui um perigo na sua prática é a sinuca. Por esses motivos é

que esse esporte não é muito praticado por crianças, pois pode ocorrer colisões das bolas com as mãos dos praticantes.

A sinuca é uma derivação do *Snooker*, que teve inicio em 1875 (BILHAR BOLA DE PRATA, 2016). Esse esporte se popularizou em diversos países, em alguns deles é um esporte nobre. No Brasil ele é praticado na maioria das vezes com 15 bolas e a mesa com 6 caçapas. As regras variam de região para região no nosso país. Muitos médicos recomendam a prática da sinuca para pessoas que tem problemas de enfartos, por se tratar de um esporte tranquilo e estratégico. Em uma partida de sinuca, os jogadores chegam a andar cerca de 1,5 quilômetros por hora (BILHAR BOLA DE PRATA, 2016).

Com todos benefícios que a interação do real com o virtual pode trazer, esse trabalho desenvolveu um jogo de sinuca virtual com a Realidade Aumentada. Com esse jogo, o usuário faz os mesmos movimentos que num jogo real, utilizando um taco de sinuca, porém em um ambiente virtual.

1.1 OBJETIVOS

Este trabalho tem como objetivo desenvolver um jogo de sinuca virtual, com interação do jogador utilizando um taco real, ou seja, um taco físico de sinuca.

Os objetivos específicos são:

a) ter um jogo de sinuca com movimentação e colisão das bolas, com placar e regras do jogo de sinuca;
b) ter um jogo de sinuca com Realidade Aumentada com interação do taco físico;
c) identificar o movimento e a posição do taco físico em uma mesa virtual, com utilização de marcadores, e interagir com o jogo.

1.2 ESTRUTURA

O trabalho está divido em quatro capítulos. O primeiro apresenta a introdução e os objetivos. O segundo capítulo apresenta a fundamentação teórica necessária para o entendimento deste trabalho. O terceiro capítulo apresenta o desenvolvimento do trabalho, com a apresentação dos casos de uso, diagramas de classes e diagramas de atividades. Também no terceiro capítulo são apresentados a implementação com a descrição das técnicas e ferramentas utilizadas durante o desenvolvimento e a operacionalidade do jogo. Ao final do terceiro capítulo é apresentado a análise dos resultados, demonstrando os resultados dos experimentos com os convidados e a comparação do jogo com os trabalhos correlatos. O quarto capítulo apresenta a conclusão do trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

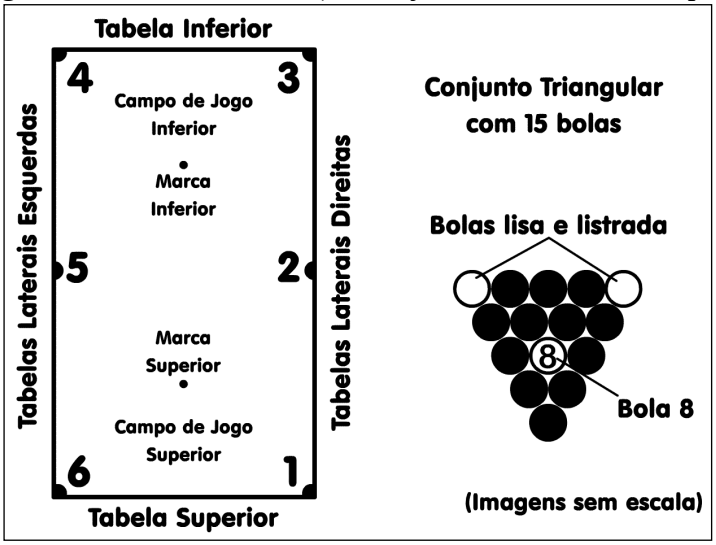
A seção 2.1 descreve as regras da modalidade Mata-8 da sinuca. A seção 2.2 apresenta o tratamento de colisão e de física. Por fim, a seção 2.3 apresenta como funciona a Realidade Aumentada e suas características.

2.1 SINUCA

A Mata-8 é uma modalidade popular da sinuca regulamentada pela Confederação Brasileira de Bilhar e Sinuca (CBBS), e que tem como objetivo converter a bola de número oito. Em sua regulamentação, algumas regras foram definidas com o objetivo de padronizar o jogo.

A regulamentação começa pela mesa, onde é definido o tamanho, marcações, tabelas e os campos. Na Figura 1 está a ilustração das marcações inferior e superior, que define a posição de partida do jogo, as tabelas, os campos e as seis caçapas numeradas. A mesa não tem um tamanho específico, porém a sua largura deve ter 50% de seu comprimento.

Figura 1 - Mesa de sinuca (Marcações, tabelas e os campos)



Fonte: CBBS (2009).

A Mata-8 deve ser jogada com uma bola branca, chamada de "tacadeira", e mais 15 bolas coloridas e numeradas. As bolas são divididas em dois grupos, de bolas lisas e bolas listradas. Grupo de bolas lisas são numeradas de 1 a 7, e as listradas de 9 a 15. A bola 8 não é considerada de nenhum grupo, e por isso se chama de bola "de jogo" (CBBS, 2009).

As bolas coloridas devem ser agrupadas em um formato triangular, conforme ilustrado na Figura 1 (lado direito), onde a bola 8 deve estar no meio do triângulo, e nos dois extremos da "base" deve estar uma bola de cada grupo. Essa base deve estar voltada para a tabela inferior e a bola do outro vértice do triângulo deve estar na marca inferior da mesa. A bola

"tacadeira" deve ser posicionada na marca superior da mesa (CBBS, 2009). Os adversários por meio de sorteio escolhem quem vai iniciar o jogo.

A tacada inicial pode visar qualquer bola do triângulo. A bola encaçapada ou convertida na tacada inicial define o grupo de bolas do jogador ativo, ficando o outro grupo para o adversário. Se for encaçapada duas ou mais bolas de diferentes grupos, o jogador ativo escolhe seu grupo de bolas e consequentemente o grupo do adversário é definido. Caso nenhuma bola seja convertida na tacada inicial, o oponente escolhe seu grupo antes da tacada seguinte. Se a bola branca for convertida, ela volta a posição inicial, na marca superior, e a jogada é passada para o adversário (CBBS, 2009).

Definido os grupos de cada jogador na tacada inicial, as próximas jogadas são intercaladas entre os jogadores, que podem continuar jogando caso converta alguma de suas bolas. Ou seja, se o jogador converter uma bola de seu grupo, ele pode realizar mais uma jogada, caso contrário é a vez do jogador adversário. A "tacadeira" deve bater primeiramente nas bolas do grupo do jogador, caso seja a bola 8 ou bolas do grupo do adversário, é considerado falta ao jogador. Se a bola branca bater na bola do jogador e depois bater na bola do adversário, não é considerado falta. Caso o jogador converta uma bola do grupo do adversário, o mesmo perde o direito de continuar jogando, caso tenha (CBBS, 2009).

A bola 8 só pode ser visada diretamente pela "tacadeira" quando todas as bolas do jogador já foram convertidas. Desta forma, a bola 8 deve ser encaçapada para o jogador ganhar a partida. Se a bola 8 for encaçapada indiretamente ou diretamente por uma jogada sem ter convertido todas do seu grupo, o jogador perde a partida (CBBS, 2009).

Todas as tacadas devem ser realizadas na "tacadeira", caso isso não seja respeitado, o jogador recebe uma falta. Essa falta, como também a falta recebida ao realizar uma jogada na bola do adversário, são punidas com a retirada de uma bola do grupo do adversário e escolhida por ele. Se o adversário estiver apenas com a bola 8, a bola é retirada e o mesmo ganha o jogo (CBBS, 2009).

A partida é finalizada após a bola 8 ser convertida ou um dos jogadores reconhecer a derrota.

2.2 TRATAMENTO DE COLISÃO/FÍSICA

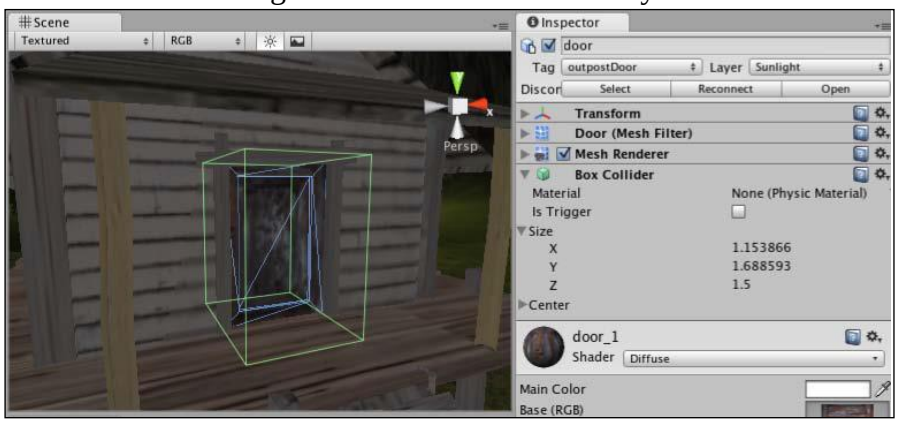
Não adianta fazer um jogo com gráficos e imagens realistas, se o objeto virtual não tiver movimentos e ações iguais ao real. Para isso existe os motores de jogos, que tem como objetivo realizar todos os cálculos e processamentos da física do objeto. Isso não é nada fácil,

pois as seguintes propriedades devem ser levadas em consideração, a massa, gravidade, velocidade e atrito (GOLDSTONE, 2009, p.13).

O Unity utiliza o motor de jogos Nvidia's PhysX, que possui uma alta precisão no seu motor de física e também é o mais popular, basta regular os parâmetros das propriedades corretamente que o motor de jogos aplicará a física sobre o objeto (GOLDSTONE, 2009, p.13). A parte mais complicada para os motores de jogos são as colisões, pois não adianta nada aplicar uma física em um carro virtual para ele andar, e quando ele bater em uma parede nada acontecer. A colisão é complicada pois alguns fatores devem ser levados em consideração, por exemplo, a força, a direção e a reação. No Unity é possível agregar a um objeto um componente *collider*, que reveste o objeto com uma rede invisível que é responsável em avisar ao motor de jogos qualquer colisão, e também com qual objeto, força e direção a colisão ocorreu. E com isso motor de jogos se encarrega de realizar a resposta dessa colisão em forma de ação (GOLDSTONE, 2009, p.13).

No Unity existem alguns tipos de colisores, os *Box Collider*, *Sphere Collider*, *Capsule Collider* e o *Mesh Collider* (UNITY TECHNOLOGIES, 2016). A diferença entre eles é a área de monitoramento de colisões que cada um tem, e com isso custo de processamento também é diferente. Segundo o manual do Unity, durante o desenvolvimento de uma aplicação os desenvolvedores devem optar na utilização dos colisores do tipo caixa, esfera e cápsula, que tem um custo baixo de processamento. Na Figura 2 está a utilização de um *Box Collider* em uma porta num jogo em Unity. Caso não seja possível utilizar esse colisores, então o desenvolvedor deve utilizar o tipo malha de colisão, onde o custo é mais alto (UNITY TECHNOLOGIES, 2016).

Figura 2 - Box Collider no Uniy



Fonte: Goldstone (2009, p. 123).

2.3 REALIDADE AUMENTADA

Essa nova tecnologia que está revolucionando o mercado, não apenas o mercado de jogos, está mudando a forma de interação das aplicações com o ser humano. Conforme Hautsch (2009), a "[...] Realidade Aumentada é uma tecnologia que permite que o mundo virtual seja misturado ao real, possibilitando maior interação e abrindo uma nova dimensão na maneira como nós executamos tarefas [...]".

O que ocorre nessa tecnologia é a sobreposição dos objetos virtuais ao ambiente real, oferecendo ao usuário uma interação multissensorial. As aplicações com RA devem ser implementadas de forma que o mundo virtual fique ajustado ao ambiente real, mesmo com o movimento do usuário. Isso deve ser seguido para passar ao usuário a sensação que o mundo virtual faz parte de seu mundo real (FAUST, 2011)

A Realidade Aumentada precisa de três componentes para funcionar. O primeiro componente é o objeto real com algum tipo de referência, que deve ser conhecida previamente pelo sistema. Essa referência pode ser um QR *Code*, uma imagem ou uma textura, isso depende da biblioteca que será utilizada para identificar a referência. Algumas bibliotecas conseguem identificar apenas QR *Code*, outras bibliotecas imagens e texturas. Câmera ou dispositivo para capturar a imagem do mundo real é o segundo componente necessário para a RA. Na imagem capturada pelo dispositivo o objeto real deve ser identificado, utilizando a referência definida. Essa identificação ocorre pela aplicação, que é terceiro componente que processa a imagem para identificar o objeto real. A aplicação também se encarrega de gerar o objeto virtual ajustado ao ambiente real, como se ambos fossem a mesma coisa, e mostrar ao usuário em algum dispositivo de saída (HAUTSCH, 2009).

As aplicações com RA podem usar diversos dispositivos de entrada e saída, mas a forma como funciona a RA não muda com os equipamentos utilizados. Podem ser utilizados na aplicação com RA uma câmera de vídeo (webcam) e um monitor de computador como dispositivos. O problema desses equipamentos em uma aplicação com pouca imersão do usuário ao ambiente aumentado (FAUST, 2011).

A utilização de óculos de realidade virtual (Head Mounted Display - HMD), oferece uma imersão muito melhor, já que é possível utilizar todo o campo de visão do usuário. Na Figura 3 está a ilustração de um HMD. Porém, alguns usuários não gostam de utilizar o HMD, por se tratar de um equipamento pesado e não ergonômico, o que causa desconforto na utilização da aplicação (FAUST, 2011).

Figura 3 - Óculos de realidade virtual (HMD)

Fonte: Nguyen (2013).

O equipamento mais utilizado nas aplicações com RA são os dispositivos portáteis, que já serve como dispositivo de entrada e saída. Muitas pessoas já possuem esses equipamentos e já estão acostamos ao dispositivo (FAUST, 2011).

Os dispositivos móveis popularizaram-se em aplicações com RA, pois ocorreram aumento na capacidade de processamento gráfico nos aparelhos atuais, o suficiente para o processamento das aplicações. Além dos avanços dos dispositivos, outro fator que contribuiu para a difusão dos aparelhos móveis, foi o aumento das larguras de banda para a transmissão de dados via internet, oferecendo aplicativos com RA colaborativos e distribuídos (FAUST, 2011).

2.4 TRABALHOS CORRELATOS

Nesta seção são apresentados três trabalhos correlatos com semelhanças aos objetivos do trabalho proposto. O primeiro é a implementação do jogo ARHockey (VIEIRA; THEODORO; TRIAS, 2006). O segundo é um jogo de sinuca com Realidade Aumentada, o Virtual Snooker (VIDA et al., 2010). E por último um jogo de boliche, o AR-Bowling (MATYSCZOK; RADKOWSKI; BERSSENBRUEGGE, 2004).

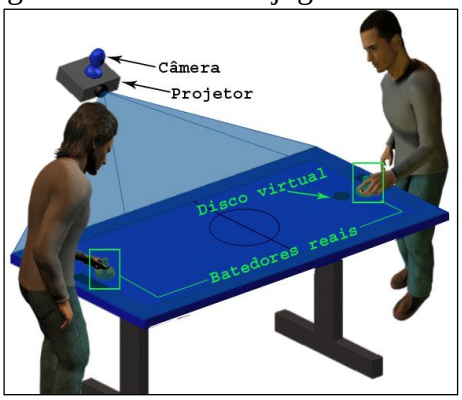
2.4.1 ARHockey

O trabalho ARHockey é a implementação do jogo airhockey, muito comum nos fliperamas, com Realidade Aumentada. Desenvolvido pelo Centro Universitário Senac de São Paulo. O jogo é constituído por duas caçapas, um disco e dois batedores, um para cada

jogador, onde o objetivo é acertar o disco na caçapa do adversário (VIEIRA; THEODORO; TRIAS, 2006).

O ARHockey foi implementado de forma que o disco do jogo e as caçapas fossem virtuais, e projetadas em uma mesa física por um projetor. Os batedores são reais e com algumas modificações para facilitar a identificação. Toda a estrutura do trabalho ARHockey pode ser visualizada na Figura 4.

Figura 4- Estrutura do jogo ARHockey



Fonte: Viera, Theodoro e Trias (2006).

Os pontos positivos levantados pelos autores é que a maneira que o jogo é projetado, todos ao redor podem ver a mesa e os movimentos dos objetos virtuais. Ou seja, não fica restrito apenas para quem está jogando. O projeto dispensa equipamentos acoplados no jogador, deixando o usuário com mais conforto para jogar.

Com a utilização de um webcam são identificados os batedores e os movimentos, e então desta forma a ação é levada ao jogo virtual. Os desenvolvedores dividiram a implementação em três subsistemas.

A primeira é a apresentação onde é ilustrado o jogo com os objetos virtuais e seus comportamentos. Essa parte foi implementada por uma *engine* 3D chamada de Ogre.

O segundo subsistema é o rastreamento, que tem como objetivo identificar os batedores com a utilização de uma câmera. Nos batedores são instalados LED infravermelhos para facilitar o processo de identificação. Desta forma, fica mais fácil a identificação por processamento de imagens. Neste subsistema o projeto identificou um problema com a utilização do webcam. O equipamento realiza o certo ajuste no brilho, contraste e tempo de exposição na captura. E o resultado foi que "[...] quando a imagem capturada é escura, a câmera aumenta o tempo de exposição, e assim a taxa de atualização das imagens cai muito." (VIEIRA; THEODORO; TRIAS, 2006). Os autores minimizaram o problema colocando o sistema de rastreamento em uma linha de execução separada.

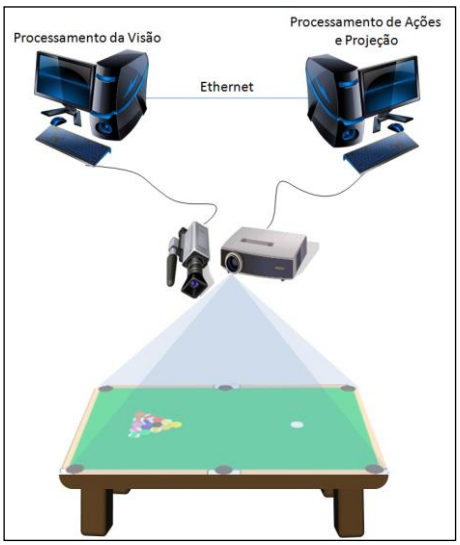
No subsistema de rastreamento, foi desenvolvido um algoritmo de visão, que tem como objetivo saber a posição dos batedores utilizando a calibração das coordenadas, onde é utilizado uma imagem quadriculada para mapear as posições. O autor ressalta que essa calibração é muito boa, pois mesmo com a câmera na posição oblíquo, é possível identificar os batedores.

Simulação Física é o último subsistema, onde é simulado todos os movimentos físicos dos objetos virtuais. A implementação foi baseada na biblioteca Open Dynamics Engine (ODE). O trabalho foi concluído com a implementação das funcionalidades básicas, como pontuação dos jogadores e a simulação de colisão. O resultado foi um projeto de baixo custo, comparado com outros trabalhos que utilizam mais equipamentos e com valor elevado.

2.4.2 Virtual Snooker

O Virtual Snooker foi um projeto do Centro Universitário da FEI, que tem a mesma estrutura que o projeto anterior, porém com um jogo de sinuca (VIDA et al., 2010). Neste projeto são utilizados dois computadores e uma câmera, essa estrutura pode ser observada na Figura 5.

Figura 5 - Estrutura do projeto Virtual Snooker



Fonte: Vida et al. (2010).

O projeto também foi dividido em três partes. A primeira é a calibragem da cor onde os desenvolvedores utilizaram uma rede neural. A segunda parte é responsável pela captura dos *frames* e identificação do marcador do taco, utilizando a rede neural, inicializada na primeira parte. A última parte é responsável pela interação física dos objetos virtuais e também da validação das regras de sinuca.

Uma das vantagens é, antes de iniciar o jogo, o usuário pode definir o tamanho da mesa da sinuca para praticar. Por motivos de iluminação do ambiente, a identificação das cores pode ter alterações, por conta disso o projeto treina uma rede neural antes de iniciar, para identificar o valor que corresponde cada cor projetada. O taco foi identificado com dois marcadores coloridos, desta forma foi possível identificar o taco e também saber a direção dele.

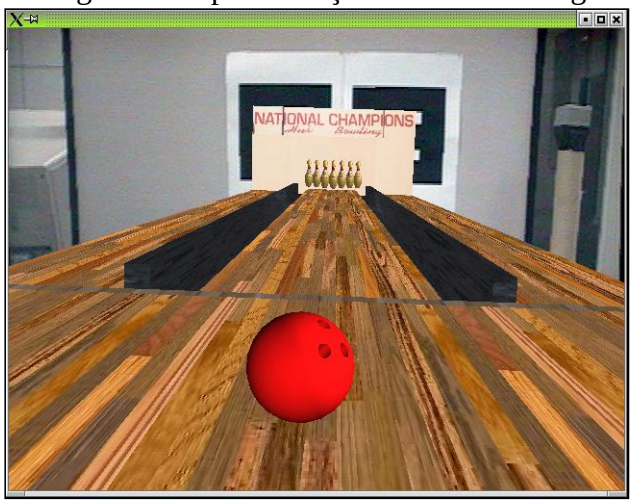
Os autores demonstraram um resultado de processamento, com uma média 91% de identificação correta do taco. Como os autores também descreveram, o projeto demonstrou ser muito robusto, pois se preocupou com a variação de cores que pode ocorrer no ambiente, porém ficou muito restrito a simulação. Não foi implementado uma interface amigável com funcionalidades para ser considerado um jogo. Como por exemplo, jogar contra um adversário e final saber o placar do jogo. Esse trabalho também não apresentou o resultado com a prática de jogadores de sinuca, para considerar o jogo realista ou não, em relação a física do jogo. Não foi usado nenhuma *engine* para simulação do jogo. O projeto Virtual Snooker também não apresentou resultado com um taco de sinuca real. Para simulação de um taco de sinuca foi utilizado um pedaço de madeira retangular, diferente do formato de um taco de sinuca.

2.4.3 AR-Bowling

O trabalho AR-Bowling, trata-se de um jogo de boliche com Realidade Aumentada. Os desenvolvedores utilizaram o *software* ARToolkit para identificação dos marcadores. Para o rastreamento da posição dos dedos, para lançamento da bola, foi usado o Fastrack da empresa Polhemus. Para simulação física foi utilizado a *engine* Vortex da CMLabs, para *engine* do jogo foi utilizado o Alchemy da Intrisic (MATYSCZOK; RADKOWSKI; BERSSENBRUEGGE, 2004).

Na figura 6 está o resultado do projeto AR-Bowling, com a Realidade Aumentada. Nesta ilustração é possível identificar o marcador usado na parede.

Figura 6- Implementação do AR-Bowling



Fonte: Matysczok,Radkowski eBerssenbruegge (2004).

O jogo de boliche funciona da seguinte forma, o usuário deve usar um dispositivo de vídeo na cabeça, que seja chama Head-Mounted Display (HMD) e uma Data Glove, conhecida como luva virtual, para identificação da posição dos dedos. Deve posicionar dois marcadores no ambiente, um no chão e outro na parede. Toda a interação do jogo é realizada fazendo os movimentos do jogo de boliche. O jogo possui as regras do boliche implementadas, como também o placar, informando a quantidade de pontos conquistados.

O autor apresentou resultados de pessoas que demoraram cerca de 1 a 2 minutos para conseguir entender e jogar o AR-Bowling. Mas ressalta que o ambiente deve ser bem iluminado e não deve ter áreas de espelhamento, para facilitar a identificação dos marcadores.

3 DESENVOLVIMENTO

As etapas de desenvolvimentos do jogo de sinuca com realidade aumentada são apresentadas neste capítulo. Na seção 3.1 são apresentados os requisitos do jogo e na seção 3.2estão os diagramas com as especificações. A implementação está apresentada de forma detalhada na seção 3.3 e na seção 3.4 o resultado do jogo obtido no projeto.

3.1 REQUISITOS

O requisito do jogo de sinuca com Realidade Aumentada são:

a) possuir uma tela inicial para seleção de opções (Requisito Funcional - RF);
b) possuir placar de bolas convertidas nas caçapas (RF);
c) exibir tempo restante para realizar a jogada (RF);
d) possuir botão virtual para recomeçar a partida (RF);
e) implementar as regras do jogo de sinuca (RF);
f) implementar os sons nas colisões das bolas (RF);
g) construir marcador do taco físico (RNF);
h) realizar a interação do taco com o jogo virtual usando o marcador (RF);
i) utilizar o *Software Development Kit* (SDK) Vuforia para a identificação do taco físico (RNF);
j) desenvolver o jogo de sinuca com o Unity (Requisito Não Funcional - RNF);
k) utilizar uma câmera para capturar a imagem a ser utilizada na identificação do taco (RNF);
l) possuir retorno visual do jogo em uma mesa com utilização de um projetor (RF);
m) utilizar a linguagem C# para definir os *scripts* do aplicativo (RNF);
n) utilizar o motor de jogos Unity para a implementação do cenário 3D do jogo (RNF).

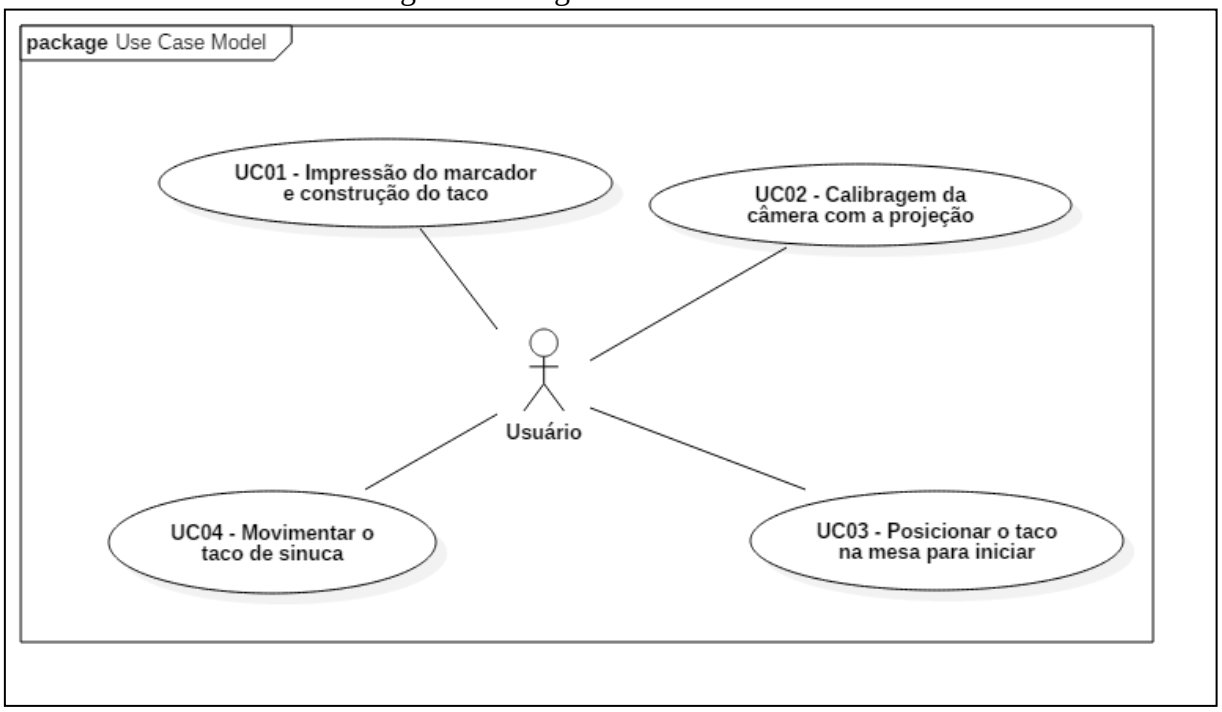
3.2 ESPECIFICAÇÃO

A especificação do jogo de sinuca foi desenvolvida utilizando a Unified Modeling Language (UML) com a ferramenta StarUML na versão 2.8.0 para o desenvolvimento dos casos de uso, do diagrama de classes e do diagrama de atividade.

3.2.1 Diagrama de Casos de uso

Nesta seção são apresentados os casos de uso do jogo, conforme a Figura 7. O jogo possui apenas o `Usuário` como ator e quatro casos de uso que serão descritos nos Quadro 1, Quadro 2, Quadro 3 e no Quadro 4.

Figura 7 - Diagrama de casos de uso



Fonte: elaborado pelo autor.

Quadro 1 - Caso de uso `UC01 - Impressão do marcador e construção do taco`

Descrição	Este caso de uso tem como objetivo disponibilizar o marcador para que seja acoplado no taco de sinuca, e também para construção do taco.
Pré-condição	Estar com os arquivos disponibilizados com o jogo, um local para impressão e um taco de sinuca ou um simulacro.
Cenário Principal	1. O `Usuário` irá imprimir a imagem `Marcador.jpg`, que está na pasta junto com o jogo; 2. Acoplar o marcador impresso na ponta do taco, ou do simulacro.
Pós-condição	O `Usuário` poderá jogar o jogo de sinuca.

Fonte: elaborado pelo autor.

Quadro 2 - Caso de uso `UC02 - Calibragem da câmera com a projeção`

Descrição	Este caso de uso tem como objetivo disponibilizar uma projeção contendo QRCodes para que o `Usuário` ajuste a câmera para capturar os QRCodes.
Pré-condição	Estar com a cena `AjusteDeCamera` aberto, projetando a imagem do aplicativo e da câmera.
Cenário Principal	1. O `Usuário` irá ajustar a câmera, analisando a imagem projetada; 2. Acoplar o marcador impresso na ponta do taco, ou do simulacro.
Pós-condição	O `Usuário` poderá jogar o jogo de sinuca.

Fonte: elaborado pelo autor.

Quadro 3- Caso de uso `UC03 - Posicionar o taco na mesa para iniciar`

Descrição	Este caso de uso tem como objetivo identificar o taco e sua altura em relação a câmera.
Pré-condição	Estar com a cena `ConfiguracaoInicial` aberta e com o taco de sinuca conforme o caso de uso `UC01`.
Cenário Principal	1. O `Usuário` deve posicionar apenas um taco sobre a mesa para que o jogo identifique o mesmo.
Exceção 1	Caso o taco com o marcador não seja identificado para configuração da altura, o jogo não é carregado.
Exceção 2	Caso o taco não seja posicionado sobre a mesa para ser identificado, pode ocorrer diferença na calibragem do taco real com o virtual.
Pós-condição	O taco pode ser retirado da mesa e o jogo de sinuca é carregado para que seja iniciado a partida.

Fonte: elaborado pelo autor.

Quadro 4 - Caso de uso `UC04 - Movimentar o taco de sinuca`

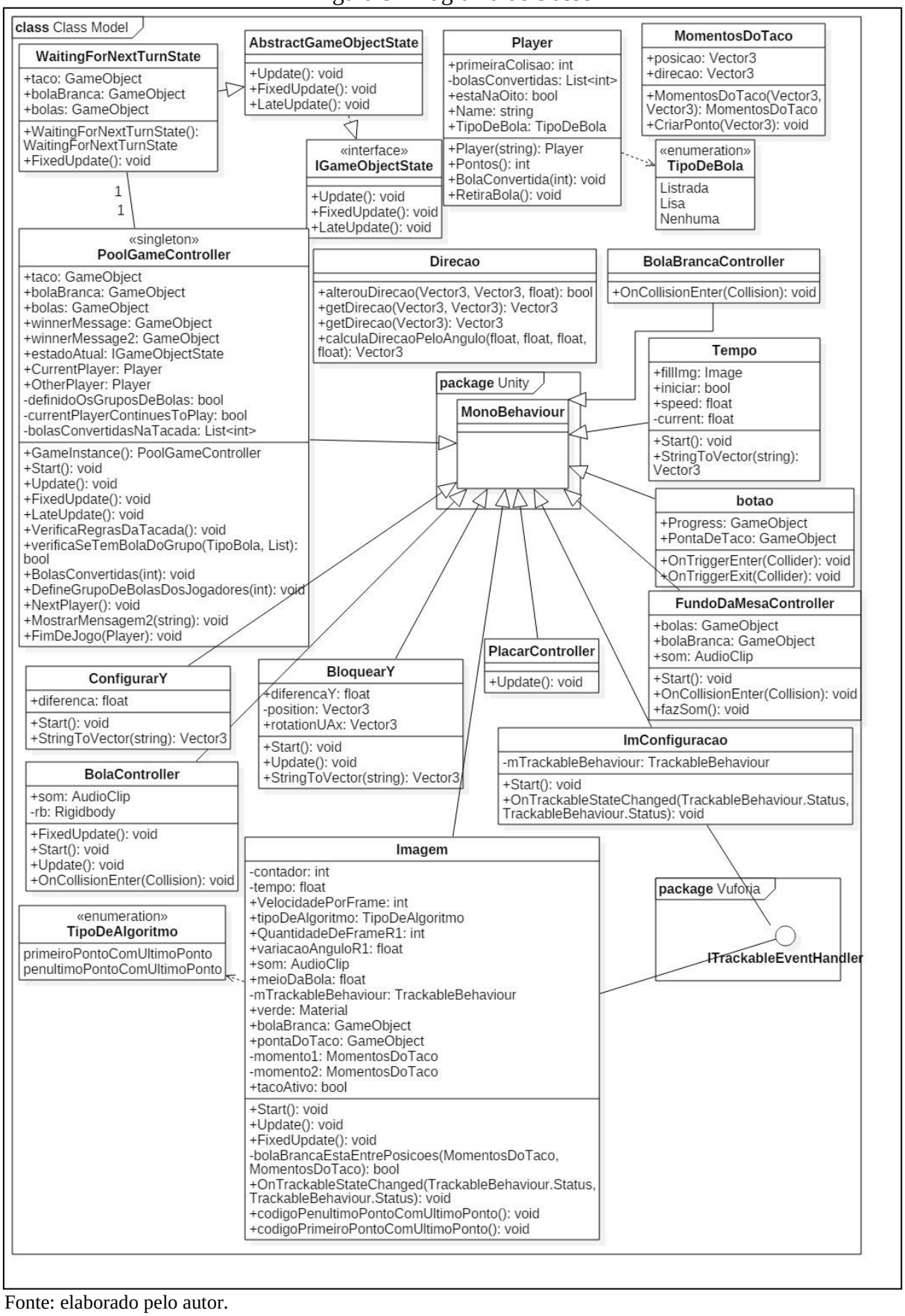
Descrição	Este caso de uso tem como objetivo utilizar o taco de sinuca ou o simulacro
Pré-condição	Ter executado o caso de uso `UC03`.
Cenário Principal	1. O `Usuário` deve utilizar o taco de sinuca para realizar tacadas na bola branca.
Fluxo alternativo	Caso o `Usuário` queira ou a partida finalize, a ponta do taco pode ser posicionada sobre o botão de reiniciar, para que seja executado o caso de uso `UC04`.
Pós-condição	Executando as tacadas na bola branca, o jogo será jogado pelo `Usuário` até finalizar a partida.

Fonte: elaborado pelo autor.

3.2.2 Diagrama de classes

As classes desenvolvidas para a criação do jogo e da cena de configuração estão apresentadas no diagrama de classes da Figura 8.

Figura 8- Diagrama de classe



Fonte: elaborado pelo autor.

As classes `Imagem` e a `ImConfiguracao` implementam o `ITrackableEventHandler`, disponibilizado pelo Vuforia para associar a um marcador. A classe `Imagem` monitora todos os movimentos realizados pelo marcador (taco de sinuca), e verifica se houve alguma colisão com a bola branca. A verificação de colisão, ou seja, de tacada, é realizado por um algoritmo que pode ser definido e pré-cadastrado na classe `TipoDeAlgoritmo`. A classe `ImConfiguracao`, que também possui um marcador vinculado, é utilizado apenas na cena de configuração da altura entre marcador e a câmera. Essa cena de configuração capta informações que serão utilizadas na classe `ConfiguraY`, para ajustar todos os objetos na altura identificada pela cena. Com essa configuração todos os objetos ficaram na altura correta em relação ao mundo real. Já a classe `BloqueарY`, é utilizada para bloquear a coordenada Y do taco, ou seja, ao mover o taco no mundo real, a coordenada Y é bloqueada pela aplicação no mundo virtual.

Todas as regras implementadas do jogo de sinuca estão na classe `PoolGameController`, que é acionado pelos demais controles quando ocorrer alguma ação no jogo. Essa classe verifica por exemplo se chegou no fim de jogo, atualiza o placar, informa o próximo jogador, entre outros. Os controles que são associados a essa classe, são os`BolaController`, `FundoDaMesaController`, `BolaBrancaController` e `PlacarController`, que controlam os respectivos objetos virtuais, as bolas de sinuca, o fundo da mesa para identificação de conversão de bola, a bola branca e o placar da partida. Por estar vinculado a objetos virtuais, todos os controles estendem a classe `MonoBehaviour` do Unity. Ou seja, quando ocorrer alguma ação nos objetos que possuem controles, a classe `PoolGameController` é acionado para verificação das regras. Para auxiliar o controle das regras da sinuca, foram implementadas as classes `Player`, `MomentosDoTaco`, `Direcao` e `TipoDeBola`, que auxiliam a verificação das regras, verificando o comportamento dos jogadores e do taco de sinuca.

O jogo possui dois estados o de liberado e o de movimento de tacada. O estado liberado é quando o taco está liberado para que o usuário possa realizar uma tacada ou acionar os botões. Quando o usuário realiza uma tacada o estado do jogo é alterado para o de movimento de tacada onde as bolas da mesa estão em movimentos, e quando o jogo está neste estado não é possível realizar novas tacadas. Para que o estado volte para o liberado, todas as bolas devem estar paradas o que significa que a tacada foi finalizada. A classe que verifica se todas as bolas estão paradas e muda para o estado de liberado é a classe `WaitingForNextTurnState`. A classe `WaitingForNextTurnState` estende uma classe base

que é a `AbstractGameObjectState`, que não possui implementação e que foi desenvolvido para manter uma orientação a objeto para caso seja desenvolvido novos estados no jogo. Além disso a `AbstractGameObjectState` implementada a interface `IGameObjectState`. Essa estrutura de classe foi mantida do projeto utilizado como base do Rehm (2015), que será explicado no capitulo 3.3.1.1.1Jogo de sinuca.

Para o desenvolvimento da interface tangível, foi implementado a classe `botao`, que associado ao um `Box Collider` para verifica se a ponta do taco entrou ou saiu da caixa de colisão. Dependendo da ação da ponta do taco na caixa de colisão, ou seja, se o taco entrou ou saiu da caixa de colisão, a rotina habilita ou desabilita o `Tempo`. Quando o `Tempo` é habilitado uma circunferência verde sobre a ponta do taco é iniciada para a contagem do tempo e o botão é acionado quando é circunferência é finalizada, ou seja, o tempo. Se a ponta do taco sair da caixa de colisão e o `Tempo` for desabilitado, a circunferência da contagem do tempo é removida e o botão não será acionado.

3.2.3 Diagrama de atividades

As funções executadas pelo usuário no jogo estão apresentadas em um diagrama de atividades na Figura 9.

Figura 9 - Diagrama de atividade do usuário

Fonte: elaborado pelo autor.

A primeira atividade que deve ser realizada pelo usuário, é a calibragem da câmera, para que a mesma consiga identificar o taco de sinuca. Antes de dar início a partida, é realizado um ajuste de altura, por isso o usuário deve posicionar o taco de sinuca sobre a mesa. Ao iniciar a partida, o jogo informa qual é o jogador que deve realizar a jogada, por isso o usuário deve aguardar a sua vez. Quando for chamado para realizar a jogada, o usuário deve movimentar o taco sobre a mesa e poderá realizar uma tacada na bola branca ou acionar o botão de reiniciar.

Se o usuário realizar uma tacada, dependendo das bolas encaçapadas e punições, o mesmo pode continuar jogando, ou seja, movimentando o taco, ou deve aguardar sua vez pois o outro jogador deve realizar sua jogada. Caso o usuário acione o botão de reiniciar, o placar é zerado e as bolas voltam para suas posições iniciais. As atividades de calibragem e configuração de altura não são necessárias ao reiniciar a partida.

3.3 IMPLEMENTAÇÃO

O processo de implementação e as ferramentas utilizadas durante o processo estão descritas na seção 3.3.1, e a operacionalidade da implementação está na seção 3.3.2.

3.3.1 Técnicas e ferramentas utilizadas

A implementação do jogo foi realizada no *framework* Unity 3D na versão 5.4.0f3 *Personal*, utilizando a linguagem de programação C Sharp para a implementação dos *scripts*. O SDK Vuforia na versão 6.2.10 para Unity foi utilizado para implementação da RA. Foi utilizado também o programa Inkscape na versão 0.48.2 para a construção e melhorias nos marcadores utilizados no projeto.

Os equipamentos utilizados em todo o desenvolvimento do projeto foi um notebook HP Pavilion dv6, com sistema operacional Windows 7 Ultimate com 6 *Gigabyte* (GB) de memória RAM e processador Intel Core i5 de 2.30 GHz, um projetor EPSON e uma webcam com 0.3 MP (mega pixels) com seis LEDs.

3.3.1.1 Inicio do projeto

O projeto foi divido em duas áreas, o desenvolvimento do jogo de sinuca e a área da realidade aumentada para identificação do movimento do taco. Essas duas áreas foram trabalhadas separadamente para isolar possíveis problemas específicos de cada área.

3.3.1.1.1 Jogo de sinuca

O projeto foi iniciado pelo jogo de sinuca, que foi desenvolvido separadamente para facilitar a implementação das regras da modalidade e sua validação. Nesta etapa foi desenvolvido um jogo de sinuca no Unity sem utilizar o Vuforia, utilizando o mouse e o teclado para jogar. Para facilitar a construção do jogo foi utilizado um projeto da matéria de computação gráfica da universidade da PUCRS realizado por Rehm (2015). O projeto de Rehm foi usado como base para o desenvolvimento do jogo, usando os objetos virtuais como a mesa e as bolas, e a estrutura do código fonte.

O projeto usado como base não havia todas as regras da modalidade implementadas, e aquelas que já estavam implementadas foram conferidas e ajustadas. O desenvolvimento se iniciou na implementação das regras, ajustando as bolas em formato triangular com a bola 8 no centro do triângulo, e posicionando na tabela inferior da mesa (Figura 1). A tacadeira foi posicionada em sua posição inicial, na marca superior da mesa.

Os jogadores já são definidos pela aplicação, e recebem o nome Jogador A e Jogador B. Os mesmos realizam tacadas intercaladas, e um jogador só pode continuar jogando caso converta uma bola de seu grupo sem levar punição. Os jogadores são informados por uma mensagem no centro da mesa, indicando qual é o jogador que deve realizar a próxima tacada (Figura 10).

Figura 10- Mesa de sinuca virtual

Fonte: elaborado pelo autor.

Cada jogador deve ter um grupo de bolas, que são as bolas lisas e as listradas. Esses grupos são definidos após um dos jogadores converter a primeira bola do jogo. O jogador que converteu a primeira bola é responsável pelo grupo da respectiva bola convertida, e o outro jogador pelo outro grupo de bolas. Caso a primeira bola seja a bola 8, o jogador que converteu perde a partida. Após definir o grupo, o placar superior informar o grupo de cada jogador, informando entre os parentes o grupo. Por exemplo "* Jogador A - 1 (Lisa) / Jogador B - 0 (List)", indica que o Jogador A é do grupo de bolas lisa, e o Jogador B é das bolas listradas.

Após definir o grupo de bolas, a primeira tacada deve ser realizada no respectivo grupo do jogador. Ou seja, se o jogador realizar uma tacada e a primeira colisão for em uma bola do grupo do adversário, o mesmo é punido pelo jogo automaticamente. Caso a tacadeira não colida em nenhuma bola, o jogador também recebe punição. Todas as punições do jogo é a retirada de uma bola do grupo do adversário, e a perda do benefício de jogar novamente caso tenha. Caso a tacada que gerou a punição converta uma bola do adversário, a punição de retirar uma bola já foi cumprida e a aplicação apenas retira o benefício de continuar jogando. Não é realizado punição se o jogador não tiver grupo de bolas definido. Em caso de punição se o adversário estiver na bola 8, a mesma é retirada e o adversário é considerado ganhador da partida. A retirada de uma bola do grupo por punição não é definido pelo jogador, e sim pela aplicação seguindo o critério da bola com o número menor entre o grupo, que não esteja convertida.

Converter a bola branca, ou seja a tacadeira, a aplicação retira o benefício de continuar jogando caso o jogador tenha, e a tacadeira retornar para a posição inicial do jogo. O benefício de continuar jogando é adquirido pelo jogador, caso o mesmo converta uma bola de seu grupo, e com isso o mesmo é informado que pode realizar mais uma tacada. Converter a bola 8 antes de converter todas as bolas de seu grupo, faz com que o jogador perca a partida. A bola 8 só pode ser visada e convertida após o jogador converter todas as bolas de seu grupo, desrespeitando essa regra o jogador recebe punição. Convertida todas as bolas de seu grupo, a conversão da bola 8 é necessária para considerar o jogador vencedor. Após a partida ser finalizada, os jogadores podem iniciar uma nova partida utilizando o botão ao lado da mesa (perto do placar de pontos).

O placar foi melhorado para indicar mais informações, como grupo de bolas de cada jogador, além de números de bolas convertidas e indicação de quem é a vez de jogar. A indicação de quem é a vez de jogar é indicada por um "*" ao lado do nome do jogador, e o mesmo também é informado por uma mensagem no centro da mesa.

Com intuito de deixar o jogo mais realista foram implementados sons nas principais ações do jogo. Os sons foram aplicados nas ações de tacada na bola branca, nas colisões entre bolas e na conversão das bolas nas caçapas. Os áudios usados para essas ações foram obtidos no projeto de sinuca desenvolvido pelo Luiz (2016).

Algumas restrições foram desenvolvidas para que a partida se desenvolva entre os jogadores. Por exemplo, a aplicação libera o jogador a realizar uma tacada apenas quando todas as bolas que estão sobre a mesa estiverem paradas. O jogo considera apenas tacadas realizadas na tacadeira, ou seja, tacadas nas demais bolas não são consideradas. A implementação dessas duas limitações melhora o desenvolvimento da partida e não gera punições desnecessárias.

A aplicação projeta uma mesa de sinuca virtual em uma posição que seja possível do jogador visualizar toda a mesa e todas as bolas. Por isso a câmera virtual foi configurada para ficar sobre a mesa e exatamente no centro, para que jogador tenha uma visão completa de todo o ambiente virtual. A imagem projetada que vai ser vista pelo jogador está ilustrada na Figura 10. Essa etapa do projeto foi finalizada com a implementação das regras da modalidade e com algumas limitações para melhorar o desenvolvimento da partida. Para que o jogo fosse testado e validado nesta etapa, a aplicação recebeu a implementação de comandos para jogar. Os comandos do jogo são as teclas de teclado direta e esquerda, para mover o taco, e o botão esquerdo do mouse para medir a força e realizar a tacada. Esses comandos foram mantidos do projeto de Rehm (2015), e nas próximas etapas foram substituídos com a implementação da realidade aumentada.

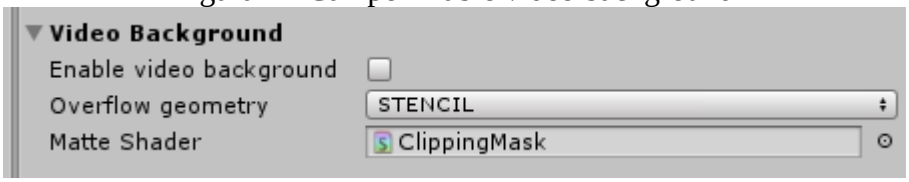
3.3.1.1.2 Realidade Aumentada

A realidade aumentada foi utilizada para identificação do taco e também a identificação do movimento realizado neste taco pelo jogador. Porém, o Vuforia não é muito utilizado para identificação de movimentos rápidos, como o que ocorre no jogo de sinuca. Por isso o trabalho iniciou nesta área com testes simples e analisando os resultados.

Durante a configuração do Vuforia foi percebido que não é necessário a visualização da imagem da câmera pelo jogador, pois a função da câmera no projeto é apenas para identificar a posição e o movimento do taco. Como essa configuração de visualização da imagem é padrão do Vuforia, a mesma teve que ser modificada neste projeto.

No objeto `ARCamera` do Vuforia, o campo `Enablevideo background` foi desmarcado, conforme Figura11. Desta forma, são projetados apenas os objetos virtuais do jogo, e a imagem captada pela câmera não é visualizada pelo usuário.

Figura11- Campo Enable video background

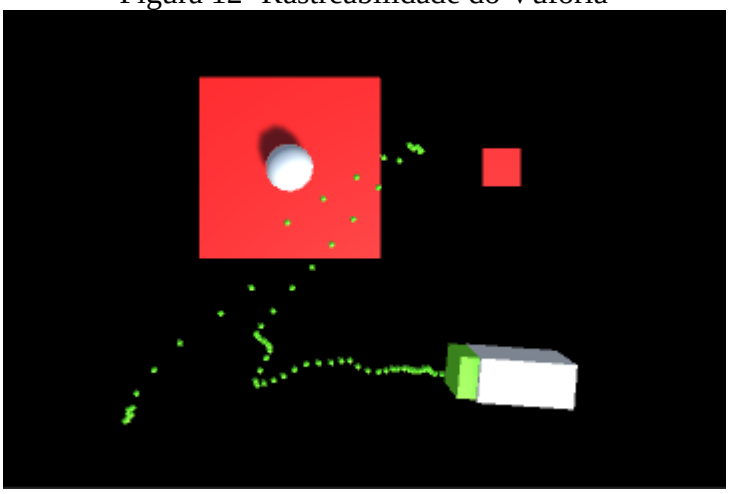


Fonte: elaborado pelo autor.

O primeiro teste do Vuforia foi verificar seu funcionamento realizando uma tacada em uma bola, sem implementação ou melhorias no algoritmo, ou seja, com as configurações padrões do Vuforia. Para isso foi desenvolvido um ambiente com uma bola sobre um plano, o que seria uma mesa, e um taco que está configurado sobre um marcador do Vuforia. Com o marcador no taco, foram realizadas tacadas na bola branca para verificar o comportamento do ambiente virtual.

Quando o marcador é movido lentamente até que seja tocado na bola, a aplicação funcionou corretamente. Porém, ao realizar um movimento mais rápido, algo comum em um jogo de sinuca, a bola não foi movida, ou seja, não ocorreu uma tacada. Para verificar o motivo desse problema, a solução foi modificada para mostrar a posição do taco em cada *frame* da aplicação. Na Figura 12 está o resultado da rastreabilidade do taco durante dois movimentos.

Figura 12 -Rastreabilidade do Vuforia



Fonte: elaborado pelo autor.

Neste teste o taco é simbolizado com um cubo branco, e um cubo verde simboliza a ponta do taco. Os pontos verdes, são posições que o Vuforia identificou o taco. O taco iniciou-se no canto esquerdo e foi movido para o centro da aplicação com uma velocidade alta, e com isso os pontos de identificação ficaram com uma grande distância entre si. Depois, o taco foi movido para baixo e para a direita, reduzindo sua velocidade até parar totalmente. Na parte final, os pontos de identificação ficaram com uma distância pequena.

A bola branca sofre o movimento de colisão com o taco apenas quando algum destes pontos é identificado na mesma posição ocupada pela bola, ou seja, algum ponto de identificação deve estar dentro da bola para realizar uma tacada. Analisando esse resultado verificou-se que quando o taco é movimentado rapidamente com o objetivo de realizar uma tacada, a distância entre os pontos é grande o suficiente para não tocarem na bola branca mesmo quando se passa o taco sobre esta bola. Ou seja, se movimentar o taco sobre a bola branca, e o primeiro ponto de identificação é antes da bola e o próximo ponto é depois, nenhum ponto de identificação está sobre a bola, e com isso a bola não sofre a tacada.

Com o teste de rastreabilidade do Vuforia, chegou-se na conclusão que não é possível o Vuforia realizar o movimento de tacada com as configurações padrões da ferramenta. Com os pontos de identificação, foi possível desenvolver um algoritmo para identificar se existe uma bola entre as posições, e caso exista o algoritmo realiza a tacada, aplicando uma força sobre a bola.

Na configuração do taco (cubo verde e branco) na aplicação, foi desabilitado o componente `RigidBody` para que não tenha a sua física aplicada sobre os demais objetos da aplicação. Desta forma, foi implementado um algoritmo que processa os pontos de identificação e verifica se a bola branca está entre os pontos, conforme o Quadro 5.

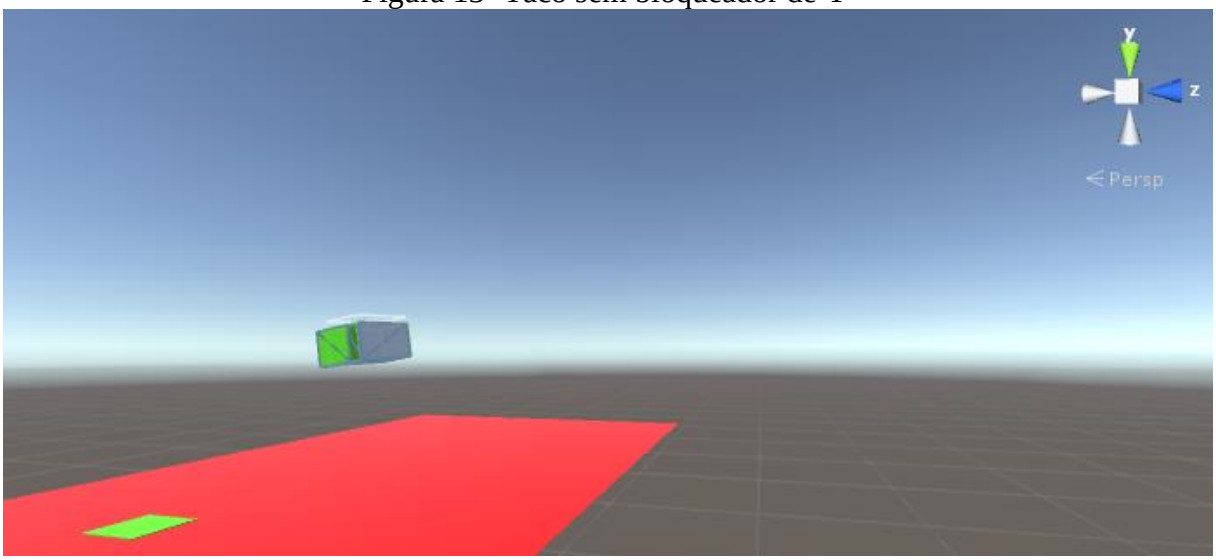
Quadro 5 - Algoritmo que verifica se ocorreu uma tacada

```
    private bool bolaBrancaEstaEntrePosicoes(MomentosDoTaco momento1,
MomentosDoTaco momento2){
        // Momento1 é a primeira posição do taco encontrado pelo vuforia
        Vector3 m1 = momento1.posicao;
// Momento2 é a ultima posição do taco encontrado pelo vuforia
Vector3 m2 = momento2.posicao;

        // Configura a altura dos pontos para o meio da bola
        m1.y += meioDaBola;
        m2.y += meioDaBola;

        // Distancia entre os dois pontos de identificação
        float dif = Vector3.Distance (m1, m2);
        // Direção realizada pelo taco entre os dois pontos
Vector3 dir = m2 - m1;
// Busca todos os objetos entre os dois pontos
        RaycastHit[] t =  Physics.RaycastAll (m1, dir, dif);

        Debug.DrawLine (m1, m1 + dir, Color.blue);
        foreach (RaycastHit f in t) {
            if (f.transform.name == bolaBranca.name) {
// Se objeto que está entre os dois pontos é a bola branca,
                // então realiza a tacada
                Debug.Log ("Tacada na bola branca - " + f.transform.name);
                return true;
            }
        }
        return false;
    }
```

Fonte: elaborado pelo autor.

Após desenvolver o algoritmo de realizar uma tacada quando um marcador passar sobre a bola branca, foi percebido um problema em relação a altura do marcador em relação ao mundo virtual, já que o Vuforia trabalha com o mundo 3D. O problema encontrado está apresentado na Figura 13, onde o usuário levantou um pouco o marcador, o mesmo não ficou na altura da mesa, ou seja, não ficou na mesma coordenada Y, e desta forma ao passar sobre a bola não é realizado uma tacada. Também na Figura 13 é possível ver que o taco está um pouco rotacionado, pois o usuário não está com o taco reto.

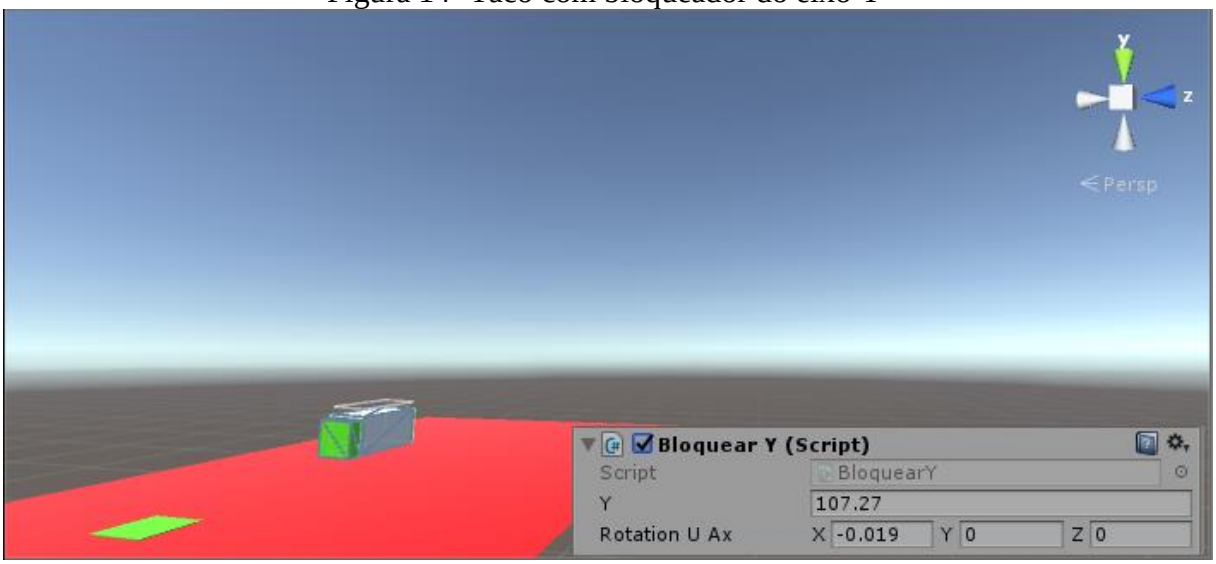
Figura 13- Taco sem bloqueador de Y



Fonte: elaborado pelo autor.

Para que o taco fique na mesma altura da mesa, e desta forma o jogador possa modificar a altura do marcador sem ter problemas, foi criado um bloqueador da coordenada Y. Além de bloquear o eixo Y do taco na mesma altura da mesa, a rotação do taco também foi bloqueado nos eixos X e Z. Desta forma, o marcador pode ficar um pouco inclinado que o taco não vai sofrer alteração no mundo virtual. Na Figura 14 está a aplicação com o bloqueador do eixo Y no taco. O marcador está na mesma posição no mundo real, que a Figura 13, e desta forma pode se ver a diferença na altura e também na rotação do taco.

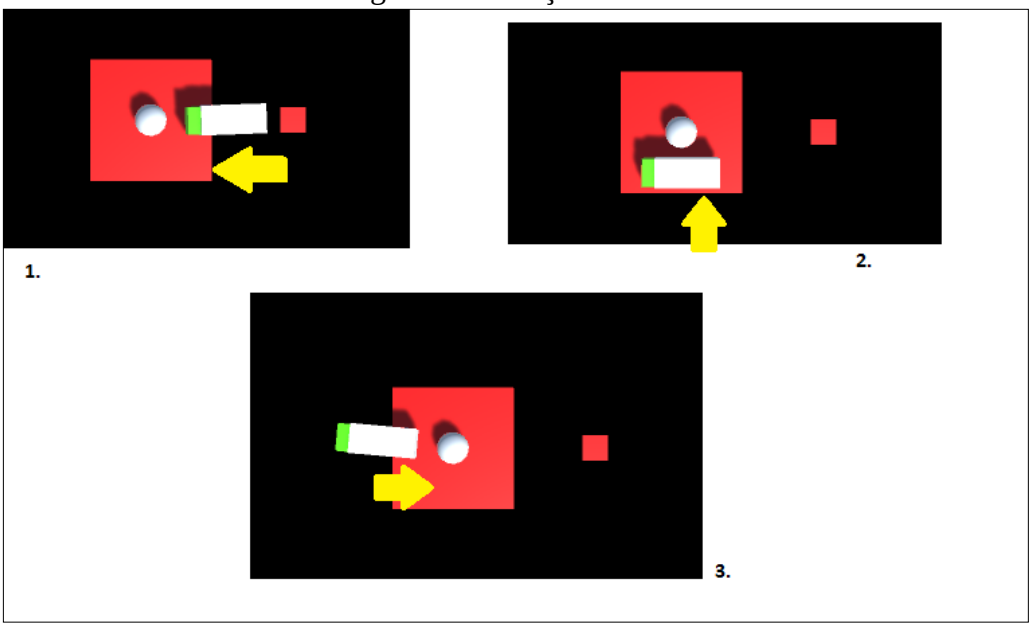
Figura 14- Taco com bloqueador do eixo Y



Fonte: elaborado pelo autor.

O taco pode passar sobre a bola branca em apenas um sentido, que é o movimento para frente no sentido da ponta do taco. As demais direções devem ser ignoradas pois não significa uma tacada na bola. Na Figura 15 está ilustrado apenas três movimentos do taco, onde apenas um pode ser considerado uma tacada.

Figura 15- Direção do taco

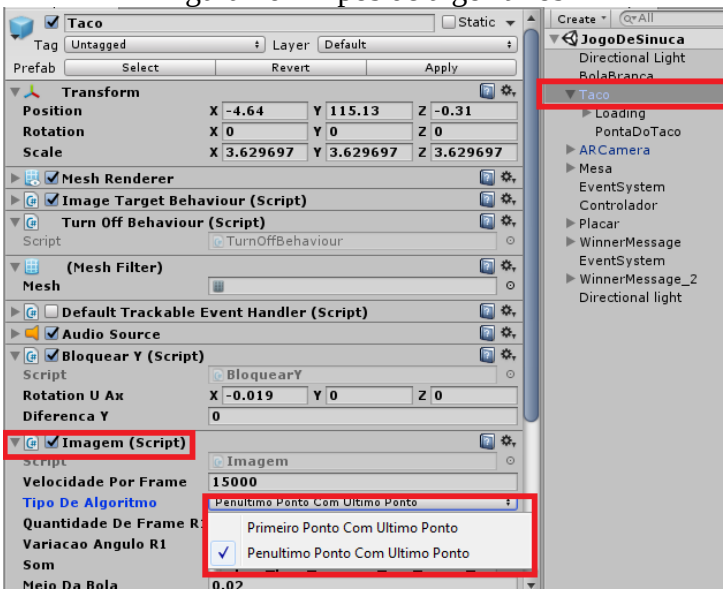


Fonte: elaborado pelo autor.

Apenas a situação 1, onde o movimentado foi para a mesma direção da ponta do taco (cubo verde) pode ser considerado uma tacada. As demais situações o movimento é para outra direção que o taco está apontado, e devem ser desconsideradas mesmo passando sobre a bola branca.

Com a implementação de verificação se a bola branca está entre dois pontos e também a da verificação da direção do taco e do movimento, foi desenvolvido dois algoritmos de tacadas. Esses algoritmos são responsáveis por calcular a força aplicada na bola dependendo da velocidade do movimento. Com os pontos de identificação do Vuforia, e com o tempo de identificação é possível calcular a velocidade em cada ponto. Para fins de testes foi implementado dois algoritmos para calcular a velocidade, e com isso ter duas formas e verificar qual é mais realista. Os algoritmos podem ser definidos pelo usuário na solução, acessando o objeto `Taco`, o componente `Imagem` e selecionado no campo `Tipo De Algoritmo`, uma das opções entre `Penultimo Ponto Com Ultimo Ponto` e `Primeiro Ponto Com Ultimo Ponto` (Figura 16).

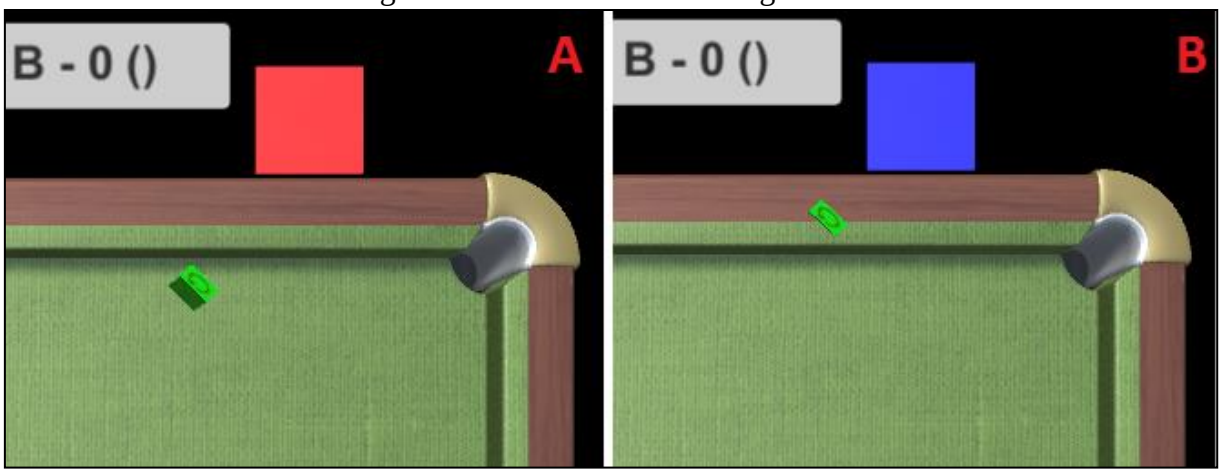
Figura 16 - Tipos de algoritmos



Fonte: elaborado pelo autor.

Durante o jogo os algoritmos podem ser modificados utilizando um botão na interface tangível. O botão se encontra abaixo do placar no canto direito da mesa e possui duas cores que indicam qual algoritmo está selecionado. O botão pode ser acionado a qualquer momento durante o jogo sem precisar reiniciar a partida. Na Figura 17 estão as duas possibilidades de cores que o botão de troca de algoritmo pode estar. No lado A com a cor vermelha significa que o algoritmo `Penultimo Ponto Com Ultimo Ponto` está selecionado e no lado B com a cor azul o algoritmo `Primeiro Ponto Com Ultimo Ponto` está selecionado no jogo. Com esse botão a configuração de algoritmo pode ser realizada pela interface do jogo sem precisar realizar a configuração da Figura 16.

Figura 17 - Botão de troca de algoritmo

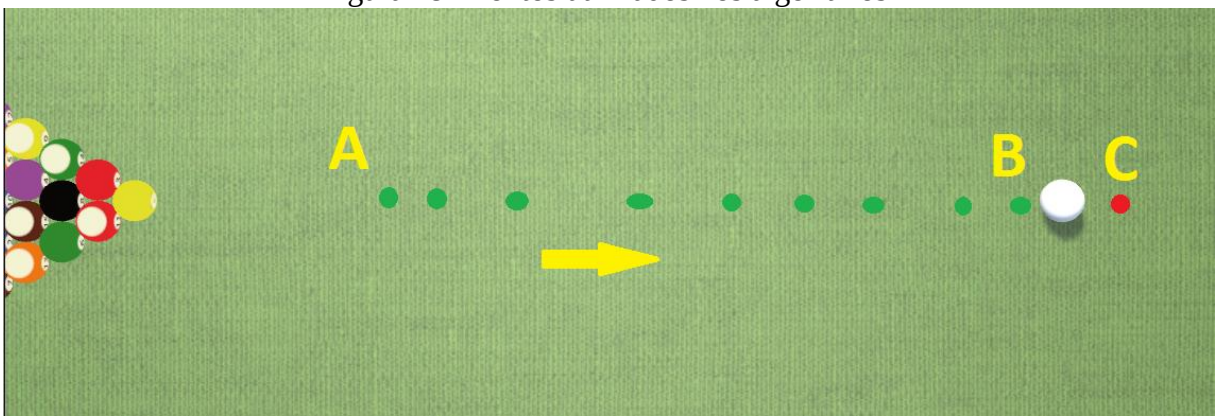


Fonte: elaborado pelo autor.

A diferença entre os algoritmos é que um trabalha com histórico de tempo e distância, e o outro utiliza as últimas informações de tempo e distância obtidas no movimento de tacada. Para facilitar a diferença, a Figura 18 ilustra pontos identificados no movimento do taco no

sentido da bola branca, saindo do ponto A até o ponto C. Conforme já visto no Quadro 5, ao identificar o ponto C, o algoritmo verifica junto com o ponto anterior (B) se a bola está entre os pontos. No exemplo da Figura 18 a bola está entre os pontos e será realizado uma tacada.

Figura 18 - Pontos utilizados nos algoritmos



Fonte: elaborado pelo autor.

Se o usuário definiu o algoritmo `Penultimo Ponto Com Ultimo Ponto`, o algoritmo calcula a velocidade utilizando o tempo e a distância obtidos nos pontos B e C, ou seja, penúltimo ponto com último ponto. No Quadro 6 e no Quadro 7 está o algoritmo que é chamado pra cada *frame* da aplicação.

Quadro 6 - Algoritmo penultimo ponto com ultimo ponto - Parte 1

```
1.  public void codigoPenultimoPontoComUltimoPonto(){
2.      contador++;
3.      tempo += Time.deltaTime;
4.      if (contador >= QuantidadeDeFrameR1) {
5.          if (tacoAtivo) {
6.              Vector3 direcaoDoMovimento = Vector3.one;
7.              if (momento1 == null)
8.                  momento1 = new MomentosDoTaco (pontaDoTaco.transform.
    position, Direcao.getDirecao (pontaDoTaco.transform.rotation.eulerAng
    les));
9.              else {
10. momento1 = momento2;
11.                 momento2 = new MomentosDoTaco (pontaDoTaco.transform.positi
    on, Direcao.getDirecao (pontaDoTaco.transform.rotation.eulerAngles));
12.                 // Direção entre os dois pontos
13.                 direcaoDoMovimento = Direcao.getDirecao (momento1.posicao,
    momento2.posicao);
14.                 bool mudouDirecaoEntrePontos = Direcao.alterouDirecao (mome
    nto1.direcao, momento2.direcao, variacaoAnguloR1);
15.                 bool mudouDirecaoDoTaco = Direcao.alterouDirecao (momento2.
    direcao, direcaoDoMovimento, variacaoAnguloR1);
16.                 if (mudouDirecaoEntrePontos || mudouDirecaoDoTaco) {
17.                     momento1 = momento2;
18.                     momento2 = null;
19.                 }
20.             ...
```

Fonte: elaborado pelo autor.

No algoritmo apresentado, entre as linhas 7 e 11, o algoritmo identifica os dois últimos pontos que serão utilizados. Entre as linhas 13 e 19 são verificados se o taco se movimentou na direção correta, conforme explicado na Figura 15. Caso tenha modificado a direção, os pontos são atualizados.

Quadro 7- Algoritmo penultimo ponto com ultimo ponto - Parte 2

```
20.             ...
21.             if (momento1 != null && momento2 != null) {
22.                 if (bolaBrancaEstaEntrePosicoes (momento1, momento2)) {
23.                     float dif = Vector3.Distance (momento1.posicao, momento
    2.posicao);
24.                     float velocidade = (dif / tempo) * 0.003f;
25.                     if (velocidade > 0.15f) {
26.                         velocidade = 0.15f; // Se não as bolas podem sair d
    a mesa
27.                     }
28.                     float force = velocidade * VelocidadePorFrame;
29.                     bolaBranca.GetComponent<Rigidbody> ().AddForce (momento
    2.direcao * force);
30.                     momento1 = null;
31.                     momento2 = null;
32.                     tacoAtivo = false;
33.                 }
34.             }
35.         }
36.         contador = 0;
37.         tempo = 0;
38.     }
39. }
```

Fonte: elaborado pelo autor.

Na linha 22 é verificado se a bola branca está entre os dois pontos, e caso esteja, entre a linha 23 e 29, é calculado a velocidade e a força, aplicando sobre a bola branca. Na linha 37 o tempo é zerado, ou seja, em cada *frame* o tempo é zerado.

Se o algoritmo definido for o `Primeiro Ponto Com Ultimo Ponto`, será considerado o tempo e a distância entre o primeiro ponto (A) até o último ponto (C), ou seja, utilizando todos os pontos. O algoritmo está apresentado no Quadro 8.

Quadro 8 - Algoritmo primeiro ponto com ultimo ponto

```
1.  public void codigoPrimeiroPontoComUltimoPonto(){
6.              ...
7.              if (momento1 == null){
8.                  momento1 = new MomentosDoTaco (pontaDoTaco.transform.
    position, Direcao.getDirecao (pontaDoTaco.transform.rotation.eulerAng
    les));
9.                  tempo = 0;
10.             } else {
11.                 momento2 = new MomentosDoTaco (pontaDoTaco.transform.
    position, Direcao.getDirecao (pontaDoTaco.transform.rotation.eulerAng
    les));
12.                 ...
16.                 if (mudouDirecaoEntrePontos || mudouDirecaoDoTaco) {
17.                     momento1 = momento2;
18.                     momento2 = null;
19.                     tempo = 0;
20.                 }
21.             }
22.             if (momento1 != null && momento2 != null) {
23.                 if (bolaBrancaEstaEntrePosicoes (momento1, momento2))
     {
24.                     float dif = Vector3.Distance (momento1.posicao, m
    omento2.posicao);
25.                     float velocidade = (dif / tempo) * 0.01f;
26.                     ...
33.  tempo = 0;
34.                     tacoAtivo = false;
35.                 }
36.             }
37.         }
38.         contador = 0;
39.     }
40. }
```

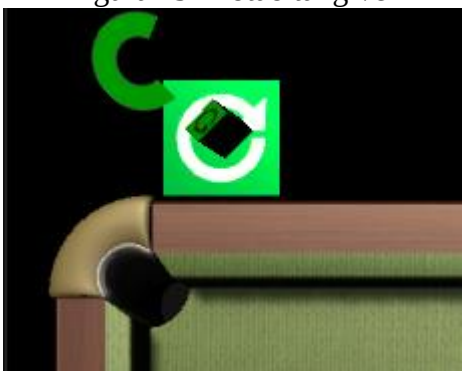
Fonte: elaborado pelo autor.

A maior diferença no código entre as duas formas de calcular a velocidade, é onde a variável `tempo` é zerada, e no momento da leitura do segundo ponto (chamado de "momento" no código). No algoritmo `Penultimo Ponto Com Ultimo Ponto` o primeiro ponto é substituído pelo segundo, mantendo os dois últimos pontos para realizar o cálculo, isso pode ser verificado no Quadro 6 linha 10. Já no algoritmo `Primeiro Ponto Com Ultimo Ponto` essa substituição não ocorre, pois, esse algoritmo trabalha como primeiro e o último ponto identificado. Essas diferenças podem ser visualizadas comparando o Quadro 6 e com o Quadro 8. São essas diferenças que fazem os algoritmos se comportarem conforme explicado na Figura 16. As demais diferenças nos códigos foram modificações para melhorar o desempenho de cada algoritmo, mas nada que altere o objetivo da implementação.

Com objetivo de não precisar utilizar equipamentos para entrada de dados como mouse e teclado, foi implementado uma interface tangível. Essa interface tangível funciona quando a

ponta do taco fica sobre o botão por um certo tempo. O tempo é indicado com uma circunferência verde, que ao completar sua circunferência, indica que o tempo foi finalizado. Na Figura 19 está a ilustração do botão com o tempo.

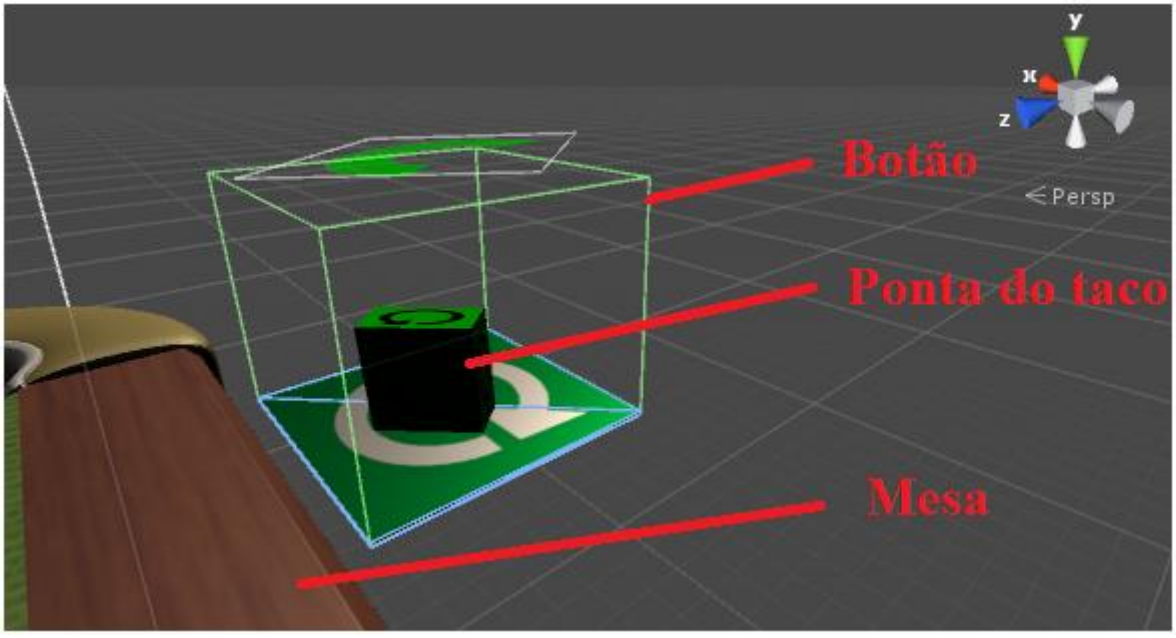
Figura 19- Botão tangível



Fonte: elaborado pelo autor.

A interface foi implementada usando `Box Collider`, que ao perceber que o taco entrou em sua área inicia a contagem do tempo. Se o usuário retirar o taco antes do tempo finalizar o botão não é acionado, ou seja, para que função do botão seja executada o usuário deve manter a ponta do taco sobre o botão até o final do tempo. Na Figura 20 o botão de reiniciar é apresentado com a ponta do taco (cubo verde) dentro de seu `Box Collider`, e com isso o tempo é iniciado sobre a ponta do taco.

Figura 20 - Botão tangível - Box Collider



Fonte: elaborado pelo autor.

No projeto foram implementados dois botões com interface tangível. O primeiro botão que está ilustrado na Figura 19 é o reiniciar, que ao ser acionado reinicia a partida da sinuca. O segundo botão é o de alteração de algoritmo, que durante o jogo o algoritmo de

identificação de pontos pode ser alterado utilizando o botão ilustrado na Figura 17. No mundo virtual do jogo, esses botões devem ficar na mesma altura da mesa e do taco, ou seja, no mesmo valor do eixo Y. Na Figura 20 é possível visualizar a base do botão tangível na mesma altura da ponta do taco e da mesa.

Para que Vuforia pudesse identificar o taco de sinuca no mundo real foi necessário desenvolver um marcador. O objetivo ao iniciar o desenvolvimento do marcador foi fazer um marcador com a mesma largura do taco. Já nos primeiros testes foi encontrado problemas de equipamentos montado para o jogo. A câmara utilizada para capturar a imagem não consegue capturar o marcador com uma qualidade que seja possível o Vuforia identificar. A distância entre o marcador e câmera não permite uma imagem nítida para que o Vuforia identifique todos os pontos de identificação do marcador. Imagens com grandes detalhes são capturadas e parecem embaçadas, impossibilitando a identificação. Já nos primeiros testes foi necessário definir que o marcador não poderia ter muitos detalhes, e por isso iniciou a utilização de QRCodes.

Para obter um marcador classificado com cinco estrelas no cadastro do Vuforia foi necessário juntar três QRCodes em um único marcador. O marcador que obteve um resultado bom de identificação pelo Vuforia foi o marcador da Figura 21, que já está acoplado no taco utilizado no jogo.

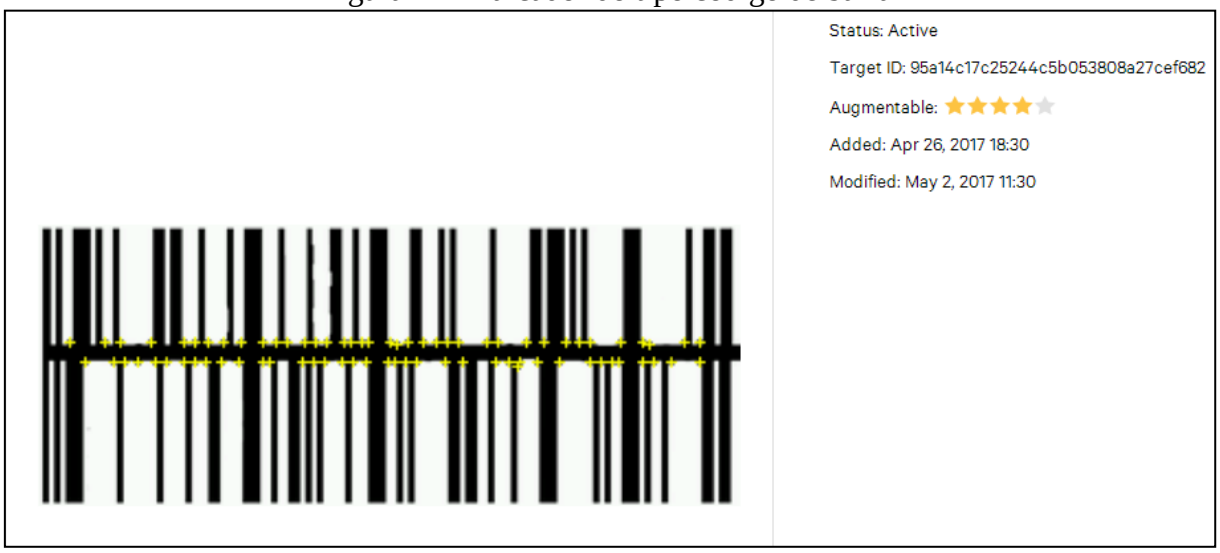
Figura 21- Taco de sinuca com marcador

Fonte: elaborado pelo autor.

A partir desse marcador tentou-se reduzir seu tamanho, para que o mesmo não atrapalhe o jogador ao realizar uma tacada. A redução do marcador não manteve o resultado satisfatório na identificação do taco, e não foi possível reduzir.

Por esse motivo, iniciou o desenvolvimento de um outro marcador utilizando a ideia de envolver a circunferência do taco com um código de barra. Envolvendo toda a circunferência do taco, o jogador poderia girar o taco que mesmo assim o Vuforia identificaria o marcador. Ao cadastrar um código de barra no site do Vuforia a classificação foi zero, pois a imagem não possui nenhum ponto de identificação com uma diferença de cor. Com o objetivo de criar pontos de identificação foi colocado uma linha no meio do código de barra, e algumas linhas do código foram removidas de um lado, para que a imagem não fique simétrica. Na Figura 22 está o marcador desenvolvido e a sua classificação de quatro estrelas.

Figura 22- Marcador do tipo código de barra

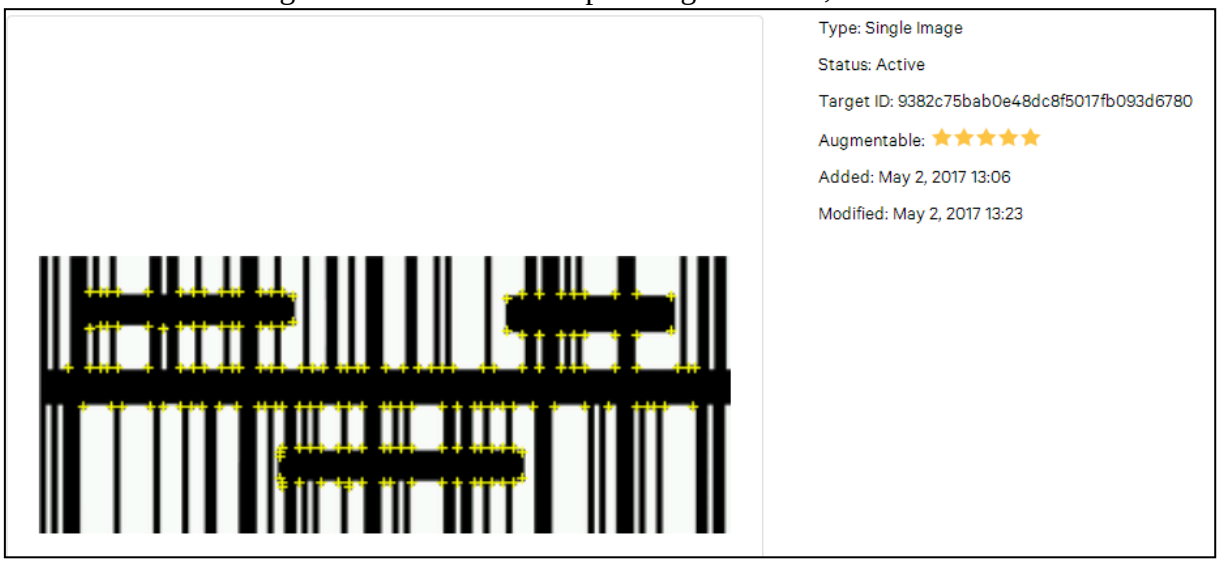


Fonte: elaborado pelo autor.

Com esse marcador, obteve-se classificação zero para uma classificação quatro, com poucas modificações. Para que o Vuforia tenha um desempenho bom na identificação é importante que o marcador tenha classificação máxima, de cinco estrelas. Utilizando esse marcador foi realizado várias modificações para que a classificação desejada seja obtida, o que não foi possível.

Realizado pesquisas e análises nos testes foi percebido que os pontos de identificação do marcador estão todos agrupados no centro da imagem horizontalmente. Para obter a classificação cinco os pontos devem estar espalhados em toda a imagem. Então com mais algumas modificações chegou-se na classificação máxima com o marcador da Figura 23.

Figura 23- Marcador do tipo código de barra, modelo 2



Fonte: elaborado pelo autor.

Comparando os marcadores com quatro estrelas com o de cinco estrelas, poucas modificações foram realizadas, e os pontos ficaram espalhados na imagem, obtendo o

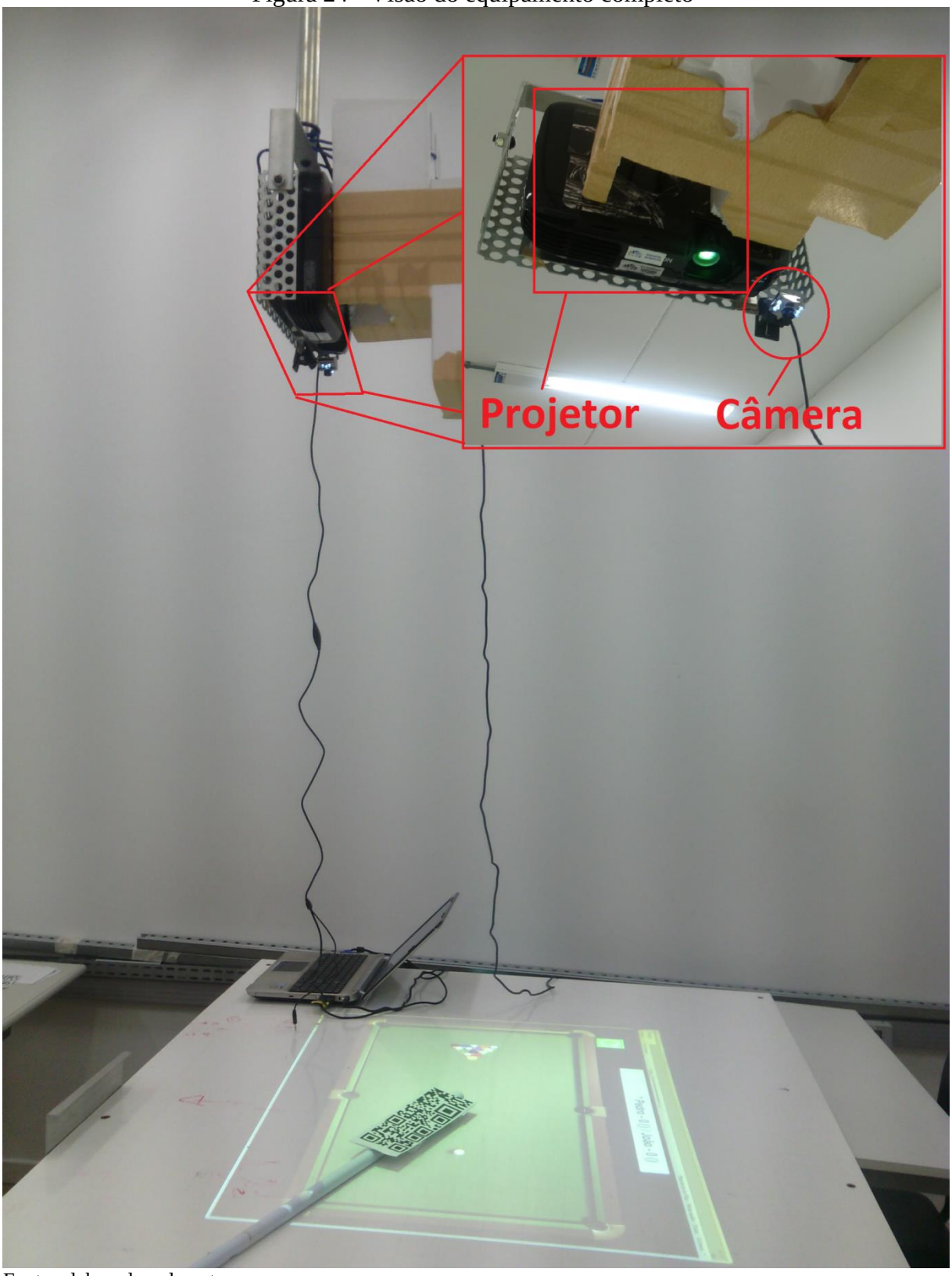
resultado esperado. Porém com o marcador da Figura 23 não é possível envolver o taco. Pois todas as linhas horizontais, que são 4 no total, devem estar visíveis a câmera ao envolver o taco, o que não é possível. Por isso, o marcador com três QRCodes, que obteve um resultado bom, será utilizado como marcador no taco de sinuca.

3.3.1.2 Incorporação da realidade aumentada no jogo de sinuca

Após as duas áreas do projeto terem chegado em resultados aceitáveis, cumprindo os requisitos, iniciou a incorporação destas áreas para fazer os testes do jogo. As ações de tacada que eram realizadas pelo teclado e mouse foram removidas, e com a incorporação da RA foi possível realizar as tacadas com o simulacro de taco de sinuca. Depois de juntar as duas áreas o equipamento foi montado nas posições corretas para iniciar os testes.

Esse projeto de sinuca virtual com RA pretende utilizar equipamentos de baixo custo e de fácil acesso, para que futuramente seja utilizado em escolas. Por isso, o equipamento utilizado não recebeu um investimento alto, como a utilização de uma câmera de alta definição. Os equipamentos eletrônicos utilizados foi um notebook, um projetor e uma câmera. O projetor foi posicionado no teto apontando sua projeção para baixo, onde foi posicionado uma mesa branca, conforme a Figura 24.

Figura 24 - Visão do equipamento completo



Fonte: elaborado pelo autor.

A câmera foi posicionada ao lado do projetor, conforme identificado na Figura 24, e ambos foram ligados no notebook que ficou sobre a mesa. Além da mesa, outro equipamento necessário é um simulacro de um taco de sinuca. Foi realizado teste com o primeiro simulacro

de taco de sinuca, que está Figura 21. Além do taco de ferro, foi desenvolvido e testado o simulacro de madeira com poucas diferenças no tamanho. Ambos são possíveis de utilizar na aplicação do jogo, e isso mostra que aplicação pode utilizar qualquer simulacro de taco de sinuca, e que não precisa de exatamente de um equipamento especifico ou de um taco de sinuca profissional. Na Figura 25 estão os dois simulacros com os marcadores, e no Quadro 9 estão as dimensões de cada simulacro utilizado. Os marcadores de ambos os simulacros estão em uma distância de 2,5 cm da ponta do taco.

Figura 25 - Simulacro de ferro e madeira

Fonte: elaborado pelo autor.

Quadro 9 - Medidas dos simulacros

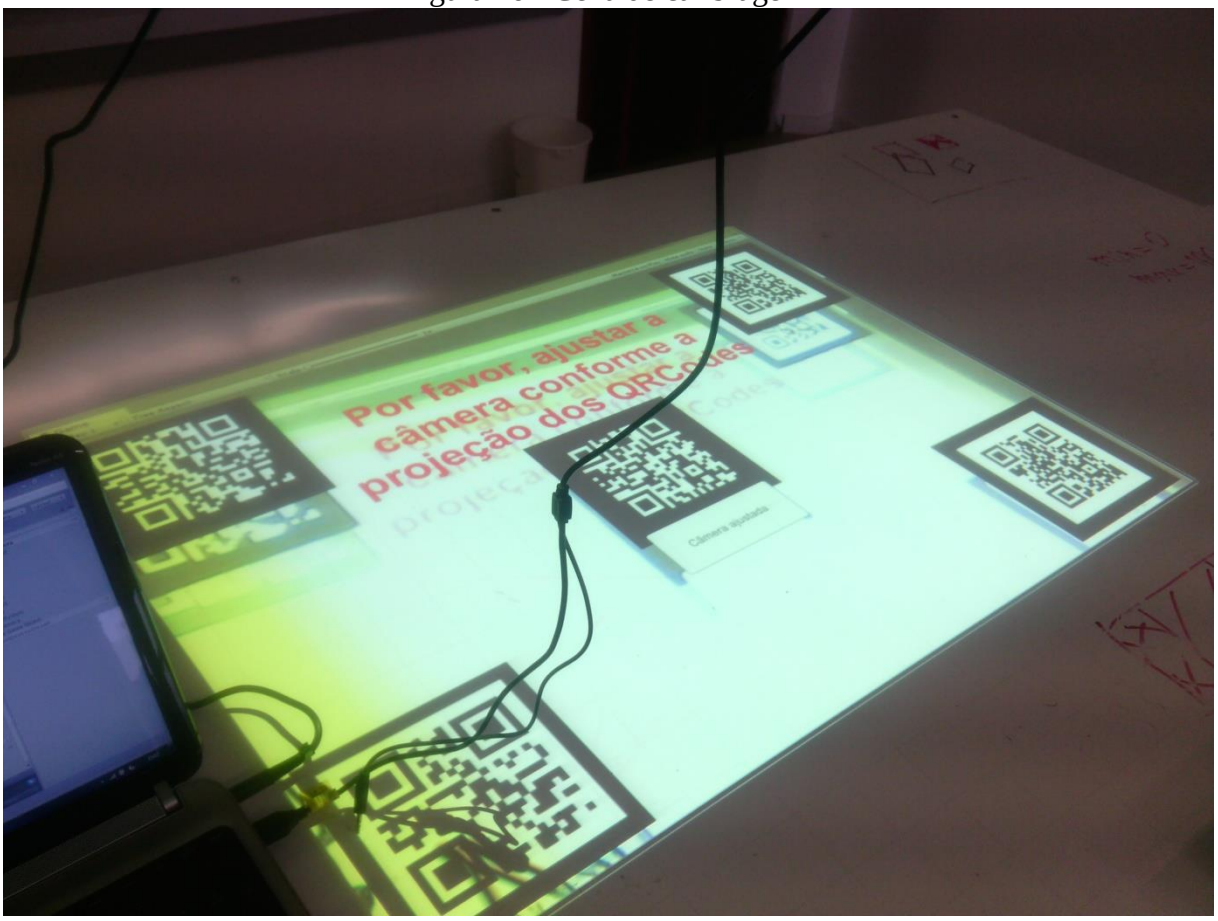
Medida	Simulacro de ferro	Simulacro de madeira
Comprimento (cm)	106	118,5
Circunferência (cm)	6	7,5
Tamanho do marcador (cm)	23 x 8	21,5 x 7

Fonte: elaborado pelo autor.

Durante a etapa de incorporação das áreas foi necessário realizar duas configurações antes do jogo iniciar. A primeira configuração é a calibragem dos equipamentos, ou seja, do projetor e da câmera. A câmera deve capturar toda a área da mesa projetada, para que o jogo consiga identificar o taco em qualquer lugar da mesa. Além disto a câmera deve estar alinhada com a projeção. Essa calibragem deve ser realizada manualmente pelo usuário, e para facilitar

foi desenvolvido uma cena que projeta cinco QRCodes que vão ajudar o usuário a calibrar a câmera. Ou seja, os cincos QRCodes devem ser capturados pela câmera e alinhados com a projeção. Na Figura 26 está a execução da aplicação de calibragem dos QRCodes.

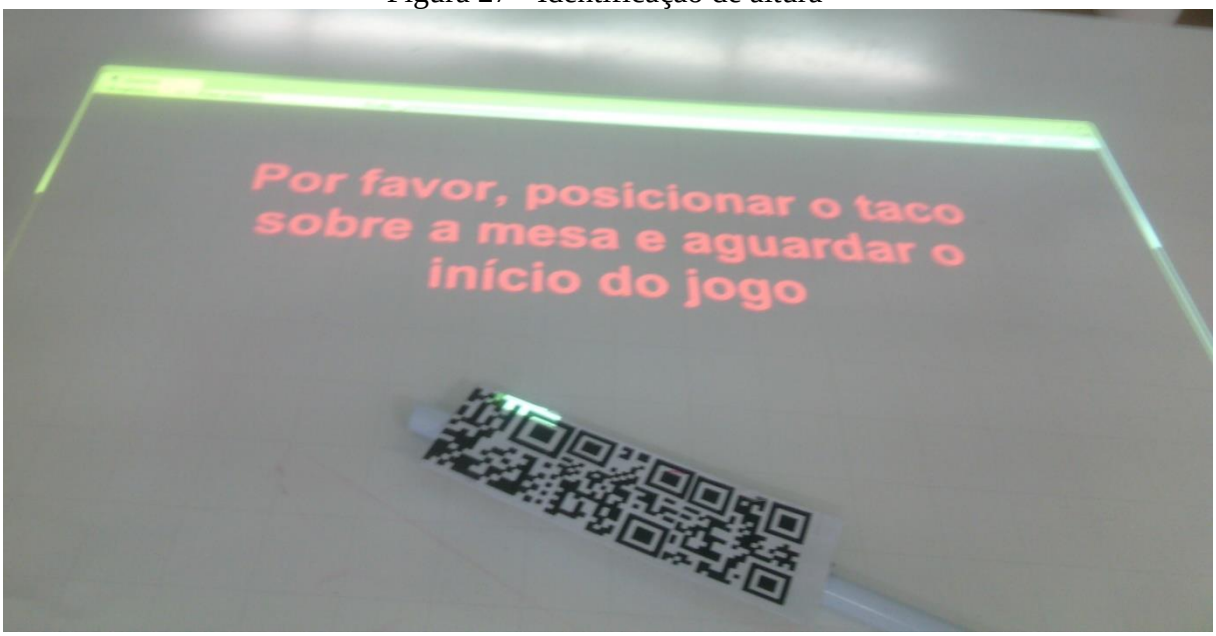
Figura 26 - Cena de calibragem



Fonte: elaborado pelo autor.

A segunda configuração é a identificação da altura da câmera com a mesa. O taco de sinuca, a mesa e todos os outros objetos virtuais, devem estar na mesma altura. Porém, a câmera pode estar em uma altura diferente, dependendo do lugar onde o equipamento será instalado. Por isso, antes de iniciar o jogo, o usuário deverá posicionar o taco de sinuca sobre a mesa para que seja realizado a leitura da altura. A altura identificada será passada como parâmetro para o jogo, e os objetos virtuais serão ajustados conforme o valor obtido. Essa configuração é realizada pela própria aplicação do jogo, ou seja, não é necessário executar outro aplicativo para realizar a configuração. Na Figura 27 está apresentada a tela solicitando o taco sobre a mesa, e o taco colocado pelo usuário. Após identificar a altura, o jogo é iniciado automaticamente.

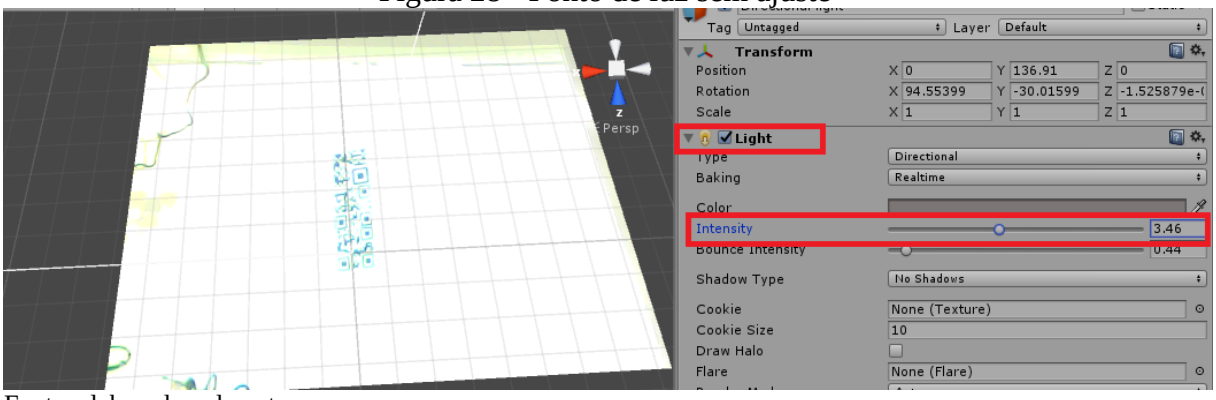
Figura 27 - Identificação de altura



Fonte: elaborado pelo autor.

O jogo de sinuca virtual possui um ponto de luz sobre a mesa para que a mesma não ficasse escura na projeção. Ao realizar a incorporação, o marcador do taco começou a apresentar problemas identificação, que ao desabilitar o ponto de luz o problema parava de ocorrer. Analisando a imagem captada pela câmera com o ponto de luz, conforme a Figura 28, foi constatado que o marcador, que está no centro da imagem, não está nítido para a identificação do Vuforia.

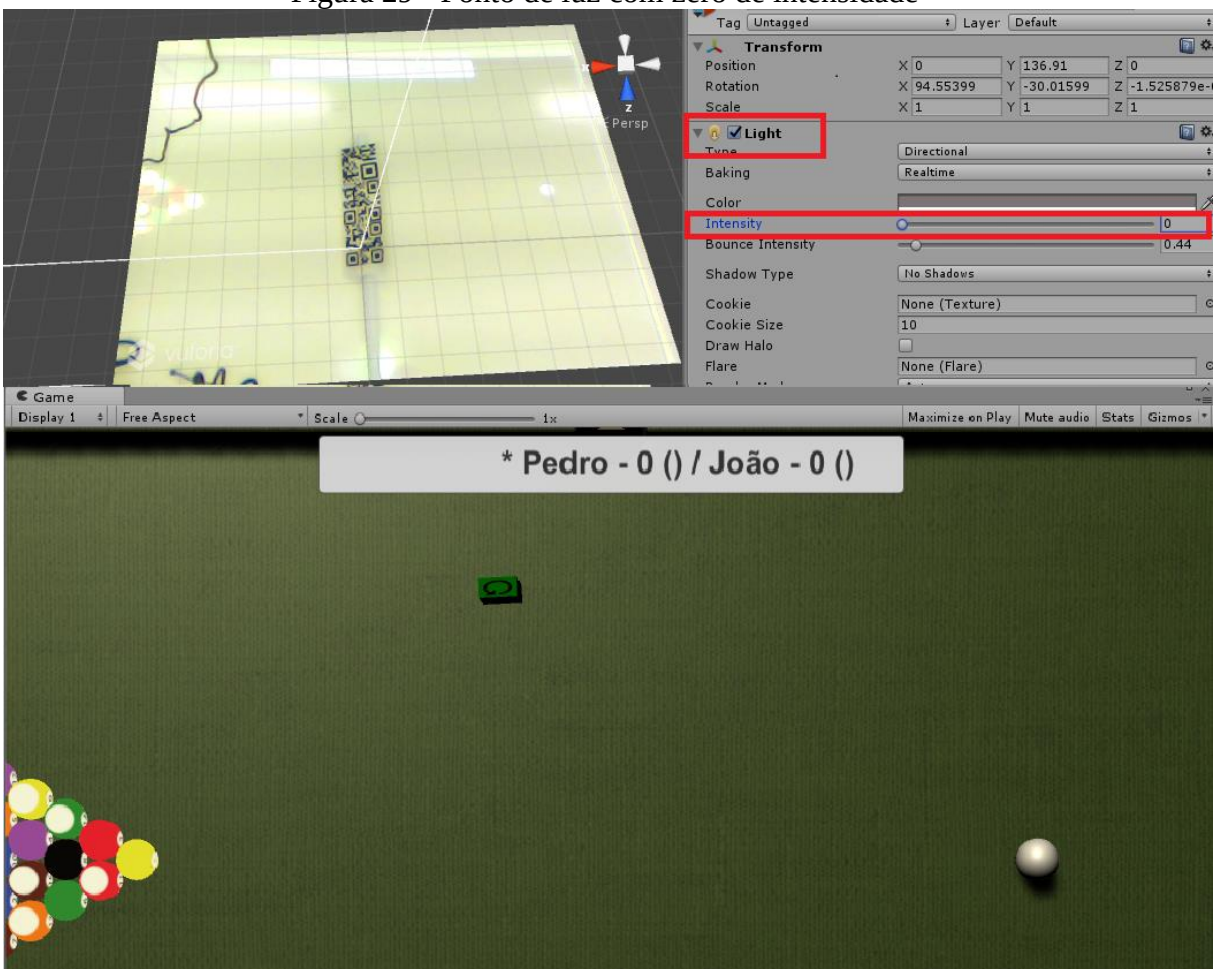
Figura 28 - Ponto de luz sem ajuste



Fonte: elaborado pelo autor.

Ao reduzir a intensidade da luz para zero, o marcador ficou nítido e o problema de identificação não ocorreu. Na Figura 29 é possível ver o marcador com nitidez, e também a mesa de sinuca projetada com a intensidade zero. A mesa de sinuca ficou muito escura ao projetar, e por isso a luz virtual é necessária.

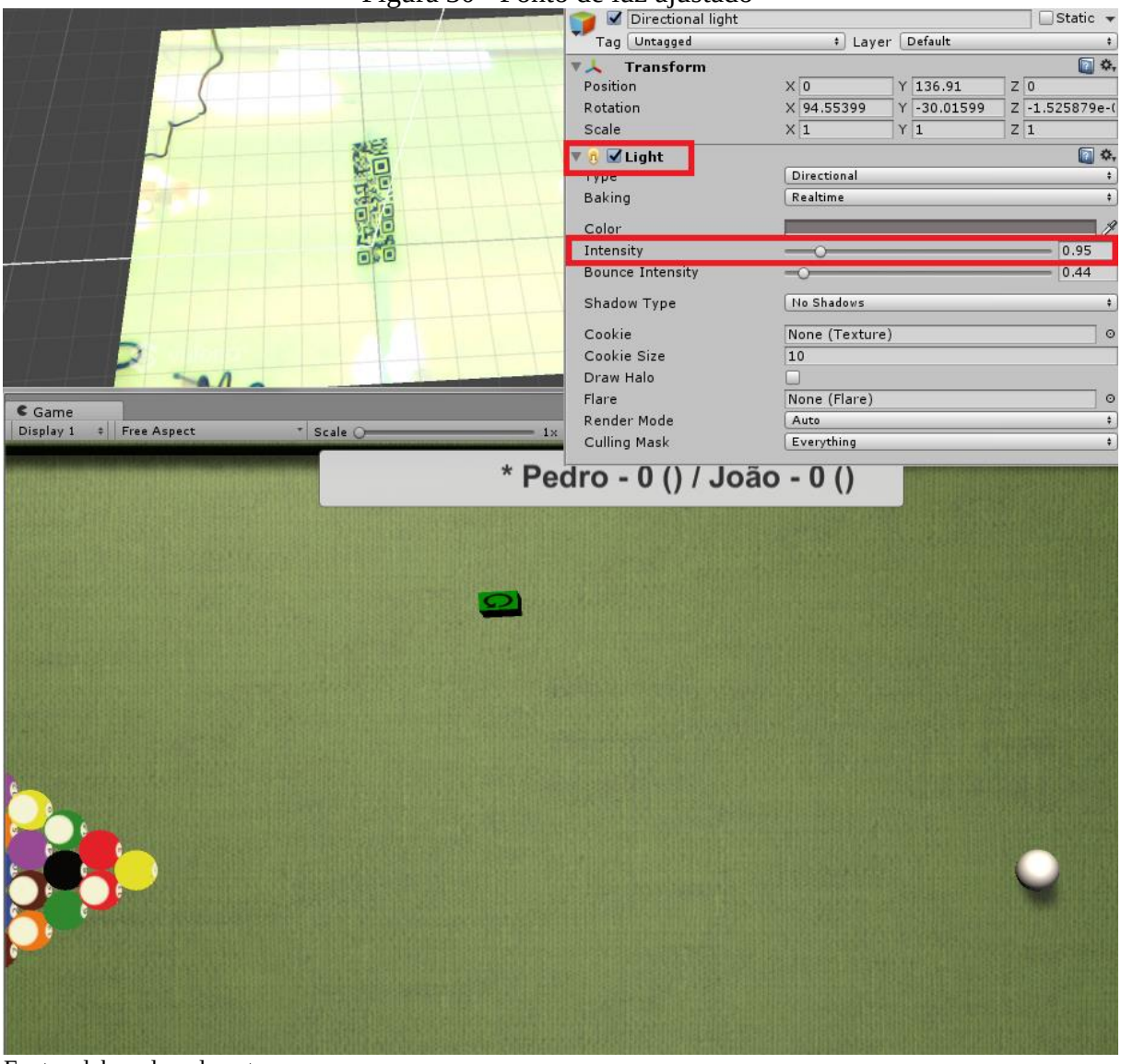
Figura 29 - Ponto de luz com zero de intensidade



Fonte: elaborado pelo autor.

Com objetivo de encontrar um meio termo para ter um pouco de luz na mesa e também para que o Vuforia consiga identificar o marcador, testes e ajustes na intensidade foram realizados e resultado está apresentado na Figura 30. Com a intensidade de 0.95, o Vuforia não tem problemas de identificação e a mesa fica iluminada o suficiente para conseguir jogar.

Figura 30 - Ponto de luz ajustado

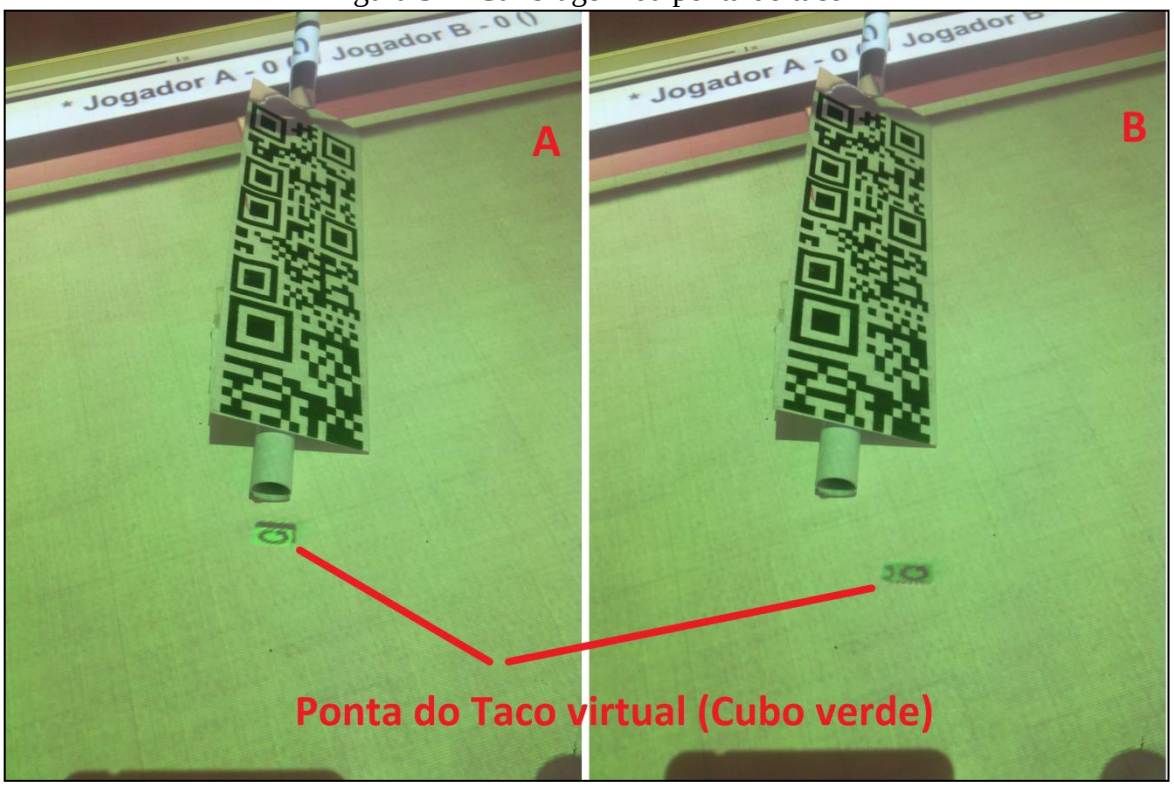


Fonte: elaborado pelo autor.

Ao identificar o simulacro do mundo real, é apresentado no jogo um cubo verde que indica a ponta do taco. Ou seja, é a posição desse cubo que é utilizado para calcular as tacadas. Se o cubo verde não passar sobre a bola branca, a tacada não será realizada.

Mesmo com a configuração de calibragem da câmera, a posição do marcador na mesa não é exatamente a mesma posição no mundo virtual, ou seja do cubo verde. Foi dado início a implementação do algoritmo que corrigi esse problema. Para realizar essa calibragem, a direção e a posição do taco na mesa, devem ser consideradas no ajuste, por isso o algoritmo é um pouco complexo e não foi totalmente implementado. Na Figura 31 estão dois casos, onde o primeiro (A) está com a ponta do taco alinhado com o cubo verde, ou seja, de forma correta e calibrada com o mundo virtual. Já no segundo caso (B), o cubo verde está ao lado da ponta do taco e mais a frente. Neste segundo caso, a rotina de calibragem não está habilitada.

Figura 31 - Calibragem da ponta do taco



Fonte: elaborado pelo autor.

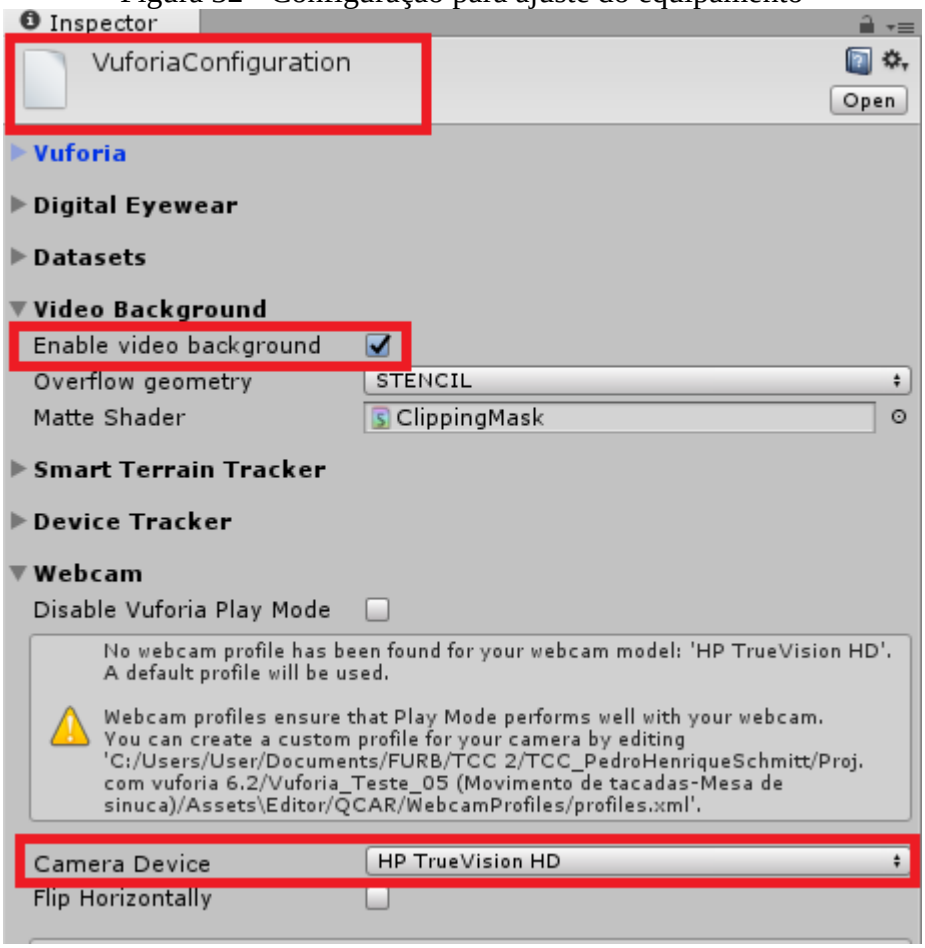
Habilitando a rotina de calibragem da ponta do taco, em alguns casos funciona corretamente, como por exemplo o caso apresentado na Figura 31, mas dependendo da direção do taco e da sua localização na mesa, a rotina não funciona. Por isso, o algoritmo não está finalizado neste projeto e deve ser continuado para a calibragem perfeita.

3.3.2 Operacionalidade da implementação

Antes de executar o jogo é necessário que o usuário imprima o marcador `Marcador.jpg`, que deve estar com os arquivos do projeto. Depois de imprimir o marcador o usuário deve adquirir ou construir um taco de sinuca ou um simulacro, e posicionar o marcador a uma distância de 2,5 cm da ponta do taco, conforme ilustrado na Figura 25.

Com uma câmera e um projetor posicionado sobre uma mesa, o usuário deve realizar a calibragem do equipamento. Para isso, o usuário deve abrir a solução do projeto, acessar a pasta `_Scenes` e abrir a cena `AjusteDeCamera`. Antes de executar a cena, o usuário deve acessar o objeto virtual `ARCamera`, clicar em `Open Vuforiaconfiguration` e realizar duas configurações. A primeira é definir a câmera que será utilizada no campo `CameraDevice` e a outra é marcar o campo `Enablevideobackgroud`, conforme a Figura 32.

Figura 32 - Configuração para ajuste do equipamento



Fonte: elaborado pelo autor.

Depois é só executar a cena para realizar o ajuste da câmera. Na projeção deverá aparecer cinco QRCodes que devem ser ajustados conforme a captação da câmera. Esse ajuste deve ser realizado pelo usuário, modificando a posição da câmera. A Figura 26 mostra a execução da cena e projeção do QRCodes com a captação da imagem pela câmera. Quanto mais ajustado for, melhor será a utilização do taco durante o jogo.

Realizada a calibragem do equipamento, o usuário deve acessar novamente as configurações do Vuforia, ou seja, acessando a `ARCamera` clicando em `Open Vuforiaconfiguration`. Na configuração do Vuforia o campo `Enablevideo background` deve estar desmarcado, ou seja, realizar o procedimento contrário ao executado na Figura 32. Depois deve ser acessado novamente a pasta `_Scenes`, e a cena `ConfiguracaoInicial` deverá ser aberta e executada.

Ao executar a tela apresentada na Figura 27 deverá ser projetada com a mensagem "Por favor, posicionar o taco sobre a mesa e aguardar o início do jogo". Neste momento o jogo realizará a identificação da altura, e por isso o usuário deverá posicionar o taco sobre

mesa, ou seja, sobre a projeção. Após posicionar o taco basta aguardar para que o jogo seja carregado.

Quando o jogo carregar, será projetada a mesa de sinuca conforme projetado na Figura 24. O jogo pode ser jogado por dois usuários, que são definidos pelo jogo como "Jogador A" e "Jogador B", e por isso os jogadores devem definir quem será cada jogador virtual. Definido os usuários, os mesmos devem aguardar a aplicação informar de quem é a vez de jogar. Essa mensagem é apresentada no centro da mesa conforme a Figura 10. Quando for a vez do jogador, o mesmo deve posicionar o taco sobre a mesa e move-la para realizar o movimento de tacada. Será projetado um cubo verde ao identificar o taco de sinuca com o marcador, esse cubo é a posição que será utilizada para captação dos pontos. Ou seja, esse cubo verde deverá passar sobre a bola branca para realizar a tacada. Em alguns momentos, esse cubo não ficara ajustado conforme o taco de sinuca.

O usuário poderá realizar uma tacada apenas quando todas as bolas estiverem paradas ou convertidas, e o jogo informar que o usuário pode jogar. Além de realizar tacadas, o jogador pode reiniciar a partida, e para isso existe um botão verde ao lado da mesa para reiniciar. Para acionar o botão o usuário deve posicionar a ponta do taco sobre o botão e aguardar o tempo de acionamento finalizar, conforme apresentado na Figura 19.

O jogo possui um placar e segue as regras da modalidade Mata-8, que tem como objetivo o usuário matar todas as bolas de seu grupo e para vencer deve converter ao final a bola 8. Os grupos de bolas são definidos após um dos usuários converter a primeira bola, e os grupos são de bolas lisas e bolas listradas. Todas as regras como punições e benefício de continuar jogando, são realizados pelo jogo e o placar superior é ajustado conforme as regras.

Quando a partida for finalizada, é apresentada uma mensagem no centro da mesa indicando quem ganhou ou perdeu. Após finalizar a partida o usuário pode iniciar uma nova utilizando o botão de reiniciar.

3.4 ANÁLISE DOS RESULTADOS

Para avaliar o jogo desenvolvido foram convidadas algumas pessoas para experimentar o jogo e algumas soluções utilizadas no desenvolvimento. Na seção3.4.1, é apresentado os experimentos realizados com os convidados e a avaliação dos mesmos. Na seção 3.4.2 é apresentado uma comparação do projeto desenvolvido em relação aos trabalhos correlatos.

3.4.1 Experimentos com convidados

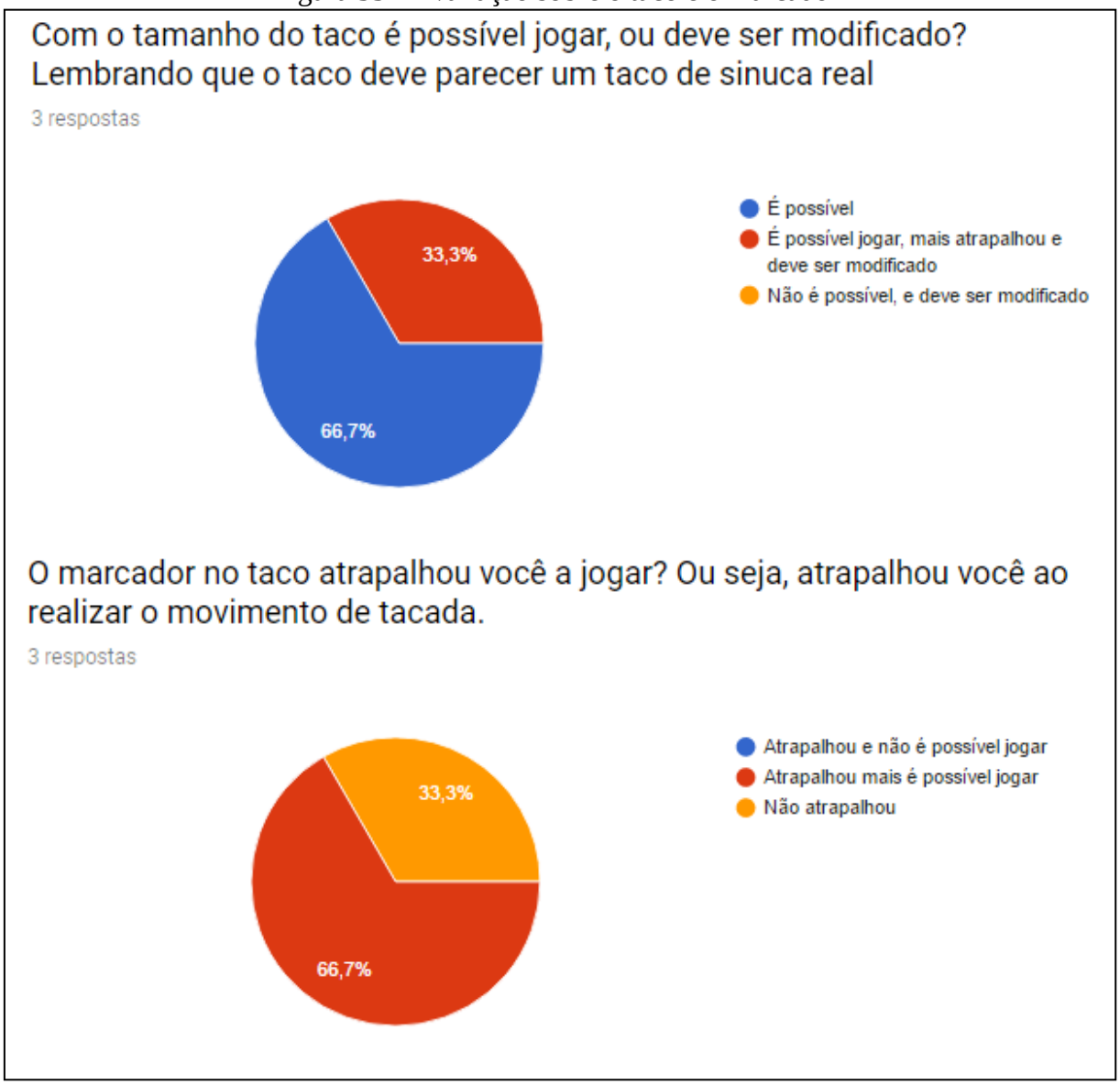
Foram convidadas para o experimento três pessoas aleatórias com intuito de avaliar e testar o jogo de sinuca. Após os testes os mesmos foram convidados a responder um questionário eletrônico, com o objetivo de identificar o tipo de usuário e a avaliação do jogo. Os usuários não receberam muitas explicações sobre o funcionamento da aplicação pois todos já conheciam as regras do jogo de sinuca e já haviam jogado a modalidade.

No questionário foi realizado perguntas para identificar os convidados, onde todos são do sexo masculino, entre 18 e 30 anos, e já jogaram sinuca e a modalidade Mata-8. Todos consideraram a ideia do jogo desenvolvido como excelente, entre as opções excelente, bom e ruim do questionário.

O primeiro experimento foi o material do taco. O projeto desenvolveu dois tacos de sinuca, um de madeira e outro de ferro. Os convidados tiveram que jogar com ambos os tacos e no questionário foi solicitado qual taco o usuário achou melhor para jogar. O taco de madeira foi escolhido por todos os convidados, alguns comentaram que o de madeira é melhor por ser mais pesado e parece com um taco de sinuca real.

Além do material do taco foi solicitado uma avaliação do tamanho dos tacos em comparação aos tacos reais, já que não é o mesmo tamanho e espessura, e também a avaliação do marcador. Na Figura 33(a), encontra-se o gráfico da resposta do questionamento e também as opções oferecidas aos convidados em relação ao tamanho do taco. Um dos usuários sentiu uma grande diferença de tamanho entre um taco real e os simulacros, porém concordou que é possível jogar com o material. Também na Figura 33 (b), está a avaliação sobre o tamanho do marcador que é acoplado sobre o taco. Não foi possível desenvolver um marcador menor e por isso foi avaliado se é possível jogar e se atrapalha no manuseio do taco durante o jogo. Dois jogadores avaliaram que o marcador atrapalha, e comentaram durante os experimentos que o tamanho do marcado oculta a projeção da mesa, e com isso prejudicando a visão do jogador. Ou seja, às vezes o marcador fica sobre a projeção da bola e o jogador não vê os objetos. Mesmo com o problema apresentado, em boa parte do jogo funcionou. Concluíram, que é possível jogar com esse tamanho de taco.

Figura 33 - Avaliação sobre o taco e o marcador



Fonte: elaborado pelo autor.

Conforme apresentado nos Quadro 6, Quadro 7 e Quadro 8 foi desenvolvido dois algoritmos para realizar a tacada, e os mesmos foram testados por todos os usuários, para avaliar a facilidade de identificação da tacada e também da força realizada no movimento. Ou seja, se a força aplicada no jogo é mais parecido com o real ou não. Neste experimento o algoritmo "`Primeiro Ponto Com Ultimo Ponto`" (Rotina 1 no questionário) foi preferido por 66,7% dos convidados, ou seja, duas pessoas. Pode-se concluir que ambos os algoritmos devem ser melhorados, e nenhum deles deve ser descartado. Em algumas situações do jogo um algoritmo é melhor que o outro, já em outras não. O que deve ocorrer em futuras melhorias e extensões é a implementação de uma inteligência artificial para decidir qual algoritmo usar em cada situação do jogo, sem que o usuário perceba. As situações mais claras

de diferença é que um algoritmo identifica melhor tacadas lentas e outro identifica melhor tacadas rápidas.

Todos os convidados gostaram de jogar o jogo de sinuca com realidade aumentada, e também consideraram que gostariam de jogar novamente. Além da satisfação dos usuários, ao comparar o jogo desenvolvido com outros jogos de sinuca virtuais, com ou sem realidade aumentada, consideraram o jogo do projeto mais realista.

O tamanho da mesa projetado ficou menor que o tamanho de uma mesa profissional de sinuca. Para saber a opinião dos usuários sobre o tamanho da mesa foi solicitado uma classificação do tamanho entre as opções excelente, bom, regular e ruim. A opção excelente recebeu um voto e a opção bom recebeu dois votos dos convidados, e por isso conclui-se que o tamanho da mesa é satisfatório para que jogo de sinuca seja praticado com os equipamentos disponíveis.

Os equipamentos utilizados não são de difícil aquisição, e foi demonstrado que é possível jogar com esses equipamentos. Talvez com uma câmera de alta definição o projeto poderia ser melhorado, como por exemplo, a redução dos marcadores, melhorando a utilização do taco pelo jogador.

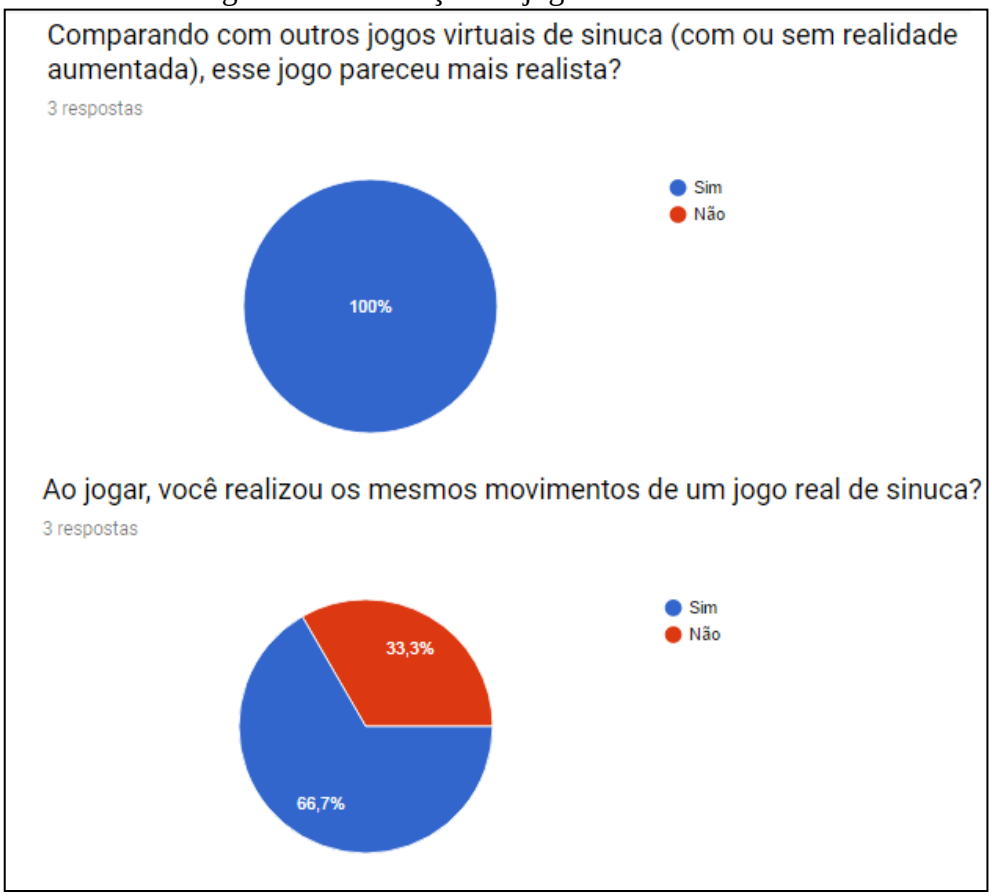
No questionário foi solicitado aos usuários que relatassem os problemas encontrados durante o jogo, e possíveis melhorias que poderiam ser implementadas. O principal problema relatado foram as jogadas nos cantos da mesa. Quando a bola branca para em algum canto da mesa, ao realizar a tacada o marcador fica posicionado fora da mesa e a câmera acaba não reconhecendo o taco e prejudicando a jogada. Se a bola estiver no meio da mesa o problema não ocorre, pois o marcador vai estar no centro da mesa, e a câmera vai conseguir capturar todo o movimento. Além desse problema, o jogo possui um *delay* entre o movimento de tacada realizado pelo jogador e a resposta do jogo no mundo virtual, porém solicitou-se aos convidados avaliarem a qualidade do *delay*. Para a pergunta "Você sentiu algum atraso (*delay*) ao realizar a tacada e ver a bola em movimento? " foram disponibilizados as opções "Sim, muito *delay* ", "Sim, pouco *delay*" e "Não". Todos os três convidados sentiram *delay* na maioria das jogadas, e um jogador sentiu pouco *delay* e os outros dois jogadores sentiram bastante *delay*. Um dos convidados comentou que o *delay* poderia ser melhorado com a utilização de equipamentos melhores, como um computador e uma câmera de alta qualidade.

No período de testes do jogo (e detalhado na seção 3.3.1.2), a intensidade da luminosidade melhorava a identificação do taco. Ao perceber os *delays* durante o experimento, foram realizados ajustes na luminosidade da sala onde estava sendo realizado o experimento, e também ajustes na luminosidade do ambiente virtual, conforme Figura 29. No

formulário dois convidados perceberam melhoras quando a intensidade da luz do ambiente virtual foi alterada para zero, da mesma maneira ilustrada na Figura 29. Os dois convidados informaram que com a intensidade zero a identificação do taco melhorou. Porém, a visibilidade da mesa piorou pois ficou bastante escuro. Levando em consideração os pontos positivos e negativos da intensidade zero, o meio termo encontrado foi de 0.95 de intensidade, que é aceitável para identificação e para visibilidade do taco, mesmo ocorrendo algumas falhas de identificação do taco.

Um dos objetivos em fazer um jogo de sinuca com realidade aumentada era de fazer um jogo mais realista em comparação aos demais jogos virtuais. E para isso o objetivo era fazer o usuário realizar os mesmos movimentos de tacada que é realizado em jogo real de sinuca. Para verificar se o trabalho conseguiu atender esses objetivos, foi realizado duas perguntas no questionário para avaliar o jogo em relação aos outros jogos de sinuca com RA ou sem RA e para verificar se o usuário realizou os mesmos movimentos de um jogo real de sinuca. Ambas as perguntas estão na Figura 34com as respostas ilustradas no gráfico.

Figura 34 - Avaliação do jogo e do movimento



Fonte: elaborado pelo autor.

Analisando os resultados das duas questões conclui-se que o jogo desenvolvido é mais realista que os demais jogos virtuais de sinuca. Ao verificar o resultado da questão do

movimento realizado pelos usuários, um convidado informou que não realizou os mesmos movimentos de um jogo real, porém dois informaram o contrário.

3.4.2 Comparação com trabalhos correlatos

Neste capítulo é apresentado uma comparação do trabalho implementado com relação aos trabalhos correlatos. No Quadro 10 é apresentado o comparativo dos trabalhos correlatos com o jogo implementado neste trabalho, que está identificado no quadro com a legenda "Sinuca com RA (2017)".

Quadro 10 - Comparação com trabalhos correlatos

Característica / Trabalhos relacionados	ARHockey (2006)	Virtual Snooker (2010)	AR-Bowling (2004)	Sinuca com RA (2017)
Equipamentos (Custo)	Câmera, projetor, computador e batedores (LED).	Câmera, projetor, dois computadores e um taco de sinuca.	Computador, HDM, luva de posição e marcadores.	Câmera, projetor, computador e um simulacro de taco de sinuca
Simulador	X	X	X	X
Utilização por 2 jogadores	X	X		X
Utiliza equipamento do esporte real	Não utiliza. Batedores foram construídos.	Não utiliza. Taco de sinuca foi construído.	Não utiliza.	Não utiliza. Foi construído com simulacro de taco.
Todos as regras do jogo real foram implementadas	X	X	X	X
Apenas quem está jogando pode visualizar o jogo			X	
Equipamentos acoplados no jogador			X	

Fonte: elaborado pelo autor.

Avaliando a comparação com os trabalhos, o projeto desenvolvido ficou mais parecido com o trabalho Virtual Snooker (2010), que também teve um taco de sinuca construído e não apresentou resultados com um taco real. A vantagem do trabalho implementado é que foi utilizado um computador a menos em relação ao Virtual Snooker, e também com uma câmera com qualidade inferior.

Todos os trabalhos não utilizaram equipamentos do esporte real, inclusive este trabalho. Porém o marcador desenvolvido para os simulacros é bem fácil de construir e de acoplar em um simulacro, e por isso concluí-se que pode ser utilizado em um taco de sinuca.

Todos os trabalhos implementaram as regras de sua modalidade e são simuladores do esporte. No jogo desenvolvido foram inseridas limitações para facilitar o andamento da partida e diminuir punições desnecessárias. Porém, essas limitações diminuem as características de ser considerado um simulador. As limitações que foram realizadas estão descritas no capitulo 3.3.1.1.2 e são o bloqueio da coordenada Y, bloqueio da rotação do taco para não poder girar-lo e o bloqueio de tacadas que não sejam realizadas para frente e apenas na bola branca. O desenvolvimento dessas limitações foi necessário para facilitar o jogo e não foi desenvolvido novas limitações para não diminuir ainda mais as características de um simulador. Mesmo com as três limitações o jogo pode ser considerado um simulador de sinuca.

O trabalho desenvolvido possui uma vantagem no requisito de equipamentos utilizados, e também não possui equipamento acoplado no jogador como no caso do trabalho AR-Bowling (2004). O jogo do trabalho AR-Bowling é visível apenas para um jogador, reduzindo a interatividade com demais jogadores. O jogo implementado é possível ser visualizado por várias pessoas, inclusive pelos jogadores.

Para jogar o ARHockey (2006) é necessário desenvolver um batedor com um *led* infravermelho. Ou seja, além do projetor, câmera e computador, é necessário de mais um equipamento extra. Esse equipamento não é muito simples de ser adquirido e construído, principalmente ao comparar com o jogo de sinuca. No jogo de sinuca deve ser utilizado um simulacro de taco de sinuca, que pode ser cabo de vassoura, e um marcador que pode ser impresso em qualquer impressora convencional.

Apenas o trabalho AR-Bowling e o trabalho realizado possuem placar para os jogadores, apenas o jogo de sinuca desenvolvido possuem sons com objetivo de deixar mas realista o jogo. Além dos requisitos já descritos, o trabalho foi desenvolvido utilizando as tecnologias Unity e o Vuforia, que são mais recentes que as tecnologias utilizadas nos trabalhos correlatos. Essas tecnologias possuem uma vantagem que é a evolução constante de ambas as ferramentas, que faz com que os projetos também evoluem.

No capítulo Operacionalidade da implementação (3.3.2) foi descrito os procedimentos que devem ser realizados na solução para a calibragem dos equipamentos e para que depois seja iniciado o jogo. Os procedimentos de calibragem não foram realizados no experimento pelos convidados, pois o equipamento já estava calibrado. Além disso nenhum trabalho

correlato menciona se é necessário realizar uma calibragem na aplicação. No jogo desenvolvido é necessária calibragem e o procedimento não é muito amigável para o usuário, pois o mesmo deve acessar várias configurações e parâmetros na solução e trocar de cenas após realizar a configuração. Não foi possível melhorar esses procedimentos pois as configurações no Vuforia na versão utilizada são salvas em um arquivo e não podem ser alteradas depois que o jogo é iniciado. Por isso, o usuário deve configurar e executar, depois parar o jogo e realizar outras configurações para iniciar novamente. Ou seja, a troca de cena automática após realizar a calibragem, para que o usuário não precise acessar a solução, não é possível. Como o procedimento de calibragem deve ser realizado apenas uma vez quando se monta o equipamento, os procedimentos não precisam ser repetidos todas as vezes que for jogar. Ou seja, se o equipamento já está montado e calibrado, o usuário pode iniciar a partida de sinuca sem precisar executar a cena de calibragem, e desta forma não é complicado para o usuário. Na seção de extensões (4.1) é apresentada uma solução para esse problema para que as configurações sejam mais amigáveis aos usuários.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um jogo de sinuca com realidade aumentada que utiliza um simulacro de sinuca para jogar. O jogo foi desenvolvido com as ferramentas Vuforia e Unity, que depois de vários testes e simulações para verificar se os movimentos de tacadas eram identificados, foi concluído que é possível utilizar essas tecnologias para o desenvolvimento do jogo de sinuca com RA.

Foram utilizados os equipamentos levantados no início do projeto que tinha como objetivo utilizar equipamentos de baixo custo. Os equipamentos utilizados foram uma câmera para identificação do taco, um projetor para projeção da mesa virtual e notebook para o processamento do jogo.

Não foi possível utilizar um taco de sinuca e por isso foi desenvolvido dois simulacros de taco que foram utilizados para interação com o jogo. Para ser identificado pelo Vuforia foi necessário o desenvolvimento de um marcador. Após vários testes chegou-se a um marcador que pode ser usado e identificado pelo equipamento utilizado, e com isso foi possível concluir mais um objetivo do trabalho.

A interação do simulacro com o jogo foi desenvolvida como uma interface tangível, onde o usuário utiliza o próprio simulacro para jogar e utilizar as funções disponíveis na interface do jogo. Um dos objetivos era possuir um botão para reiniciar a partida, com a interface tangível foi possível desenvolver o botão de reiniciar e também outro botão para a função de troca de algoritmo.

O jogo virtual foi implementado com várias características que atenderam a maioria dos objetivos, como um placar de bolas convertidas que também indica o grupo de bolas dos jogadores, a implementação de todas as regras da modalidade Mata-8 com as punições por falta e aplicação de sons nas principais ações do jogo para deixar mais realista.

O objetivo de exibir um tempo restante para o jogador realizar a jogada não foi desenvolvido para que o jogo tenha características de simulador, e a modalidade não possui tempo limite para realizar jogadas. Com isso, o jogador tem o tempo que achar necessário para realizar a tacada. O requisito de possuir uma tela inicial para seleção de opções não foi necessário para o jogo desenvolvido, pois não foi implementado opções para o jogador utilizar o jogo e as configurações necessárias foram colocadas na interface tangível.

Mesmo tendo várias coisas para melhorar, principalmente na calibragem da ponta do taco e na redução dos *delays*, o trabalho conseguiu ser desenvolvido e pode ser utilizado para

realizar uma partida de sinuca com RA. Isso foi possível comprovar no questionário realizado por convidados que avaliaram o jogo.

Por fim, o trabalho se mostrou de grande valia, pois pode ser utilizado na aprendizagem de pessoas interessadas no esporte. Também se mostra importante para futuras extensões e para trabalhos semelhantes que pretendem identificar movimentos, pois as tecnologias utilizadas são atualizadas e não foram utilizadas por outros trabalhos para esse propósito. Por isso, esse trabalho possui um legado com testes, simulações e desenvolvimentos de algoritmos com RA para identificação de movimentos.

4.1 EXTENSÕES

Para futuras extensões do trabalho, as seguintes sugestões são:

a) continuação das pesquisas e desenvolvimentos dos marcadores para redução de seu tamanho com o objetivo de envolver o taco;
b) utilização de melhores equipamentos, principalmente da câmera;
c) aumento da projeção para que a mesa fique maior e parecida com uma mesa de sinuca profissional;
d) continuação do algoritmo de calibragem da ponta do taco;
e) redução dos *delays*, para isso o algoritmo e os equipamentos devem ser melhorados;
f) desenvolvimento de dois aplicativos para que o usuário não tenha que trocar de cenas e alterar configurações para realizar a calibragem, ou seja, para deixar mais amigável ao usuário. Para isso um aplicativo faz a calibragem e o outro executa o jogo de sinuca;
g) atualizara identificação de tacadas para permitir tacadas com efeito, algo realizado por profissionais que jogam sinuca;
h) implementação de uma inteligência artificial para definir qual o algoritmo de tacada deve ser utilizado em cada situação do jogo;
i) melhorias na identificação de tacadas no canto da mesa, onde o marcador fica fora da área de captura da câmera;
j) cadastro de usuário para informar o nome no placar e também para manter um histórico de jogadores;
k) utilização de multi-câmeras e câmera 3D.

REFERÊNCIAS

BILHAR BOLA DE PRATA. **Curiosidades**. Campinas,[2016?]. Disponível em: <http://www.bilharboladeprata.com.br/curiosidades/>. Acesso em: 10 ago. 2016.

CONFEDERAÇÃO BRASILEIRA DE BILHAR E SINUCA. **Regras Populares**: Mata-8. [s. L.], 2009. 3 p.

FAUST, Fernanda G. et al. Aplicações e Tendências da Realidade Aumentada no Desenvolvimento de Produtos. In: CONGRESSO BRASILEIRO DE GESTÃO DE DESENVOLVIMENTO DE PRODUTO, 8., 2011, Porto Alegre, RS.**Anais...**Porto Alegre: UFRGS, 2011. p. 1-10.

FERREIRA, Emmanoel. Games e Imersão: a realidade híbrida como meio de imanência virtual. In: SIMPÓSIO NACIONAL DA ABCIBER, 2., 2008, São Paulo. **Anais...** São Paulo: ABCiber, 2008. p. 1-15.

FINCO, Mateus D.; FRAGA, Alex B. Rompendo fronteiras na Educação Física através dos videogames com interação corporal. **Motriz**, v.18 n.3, p.533-541, jul./set. 2012.

FOLHA ONLINE. **Nintendo Wii é o equipamento que faz as pessoas mais felizes na Inglaterra, diz pesquisa**. [S.l.], 2009. Disponível em: <http://www.mauavirtual.com.br/noticia-10720.html>. Acessoem: 28 ago. 2016.

GOLDSTONE, Will. **Unity Game Development Essentials**. Birmingham, Mumbai: Packt Publishing, 2009. Disponível em: <http://www.enucomp.com.br/2012/conteudos/minicursos/unity.pdf>. Acesso em: 13 set. 2016.

LUIZ, Ricardo. **Jogo de Sinuca Feito com Unity.** [S.l.], 2016. Disponível em: <https://github.com/ric-luiz/sinuca>. Acessoem: 15 dez. 2016.

MATYSCZOK, Carsten; RADKOWSKI, Rafael; BERSSENBRUEGGE, Jan. AR-Bowling: Immersive and Realistic Game Play in Real Environments Using Augmented Reality. Conference: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology,1., 2004, Singapore.**Proceedings…**Singapore: ACM, 2004. p. 269-274.

NGUYEN, Vincent. **Canon MREAL Mixed Reality Hands-on**. [S.l.], 2013. Disponível em: < http://www.slashgear.com/canon-mreal-mixed-reality-hands-on-21270443/>. Acesso em: 31out. 2016.

REHM, Fabio. **Unity 3D Pool.** Rio Grande do Sul, 2015. Disponível em: <https://github.com/fgrehm/pucrs-unity3d-pool/blob/master/README.md>. Acesso em: 20 dez. 2016.

UNITY TECHNOLOGIES. **Unity Manual**. [S.l.], 2016. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em: 13 set. 2016.

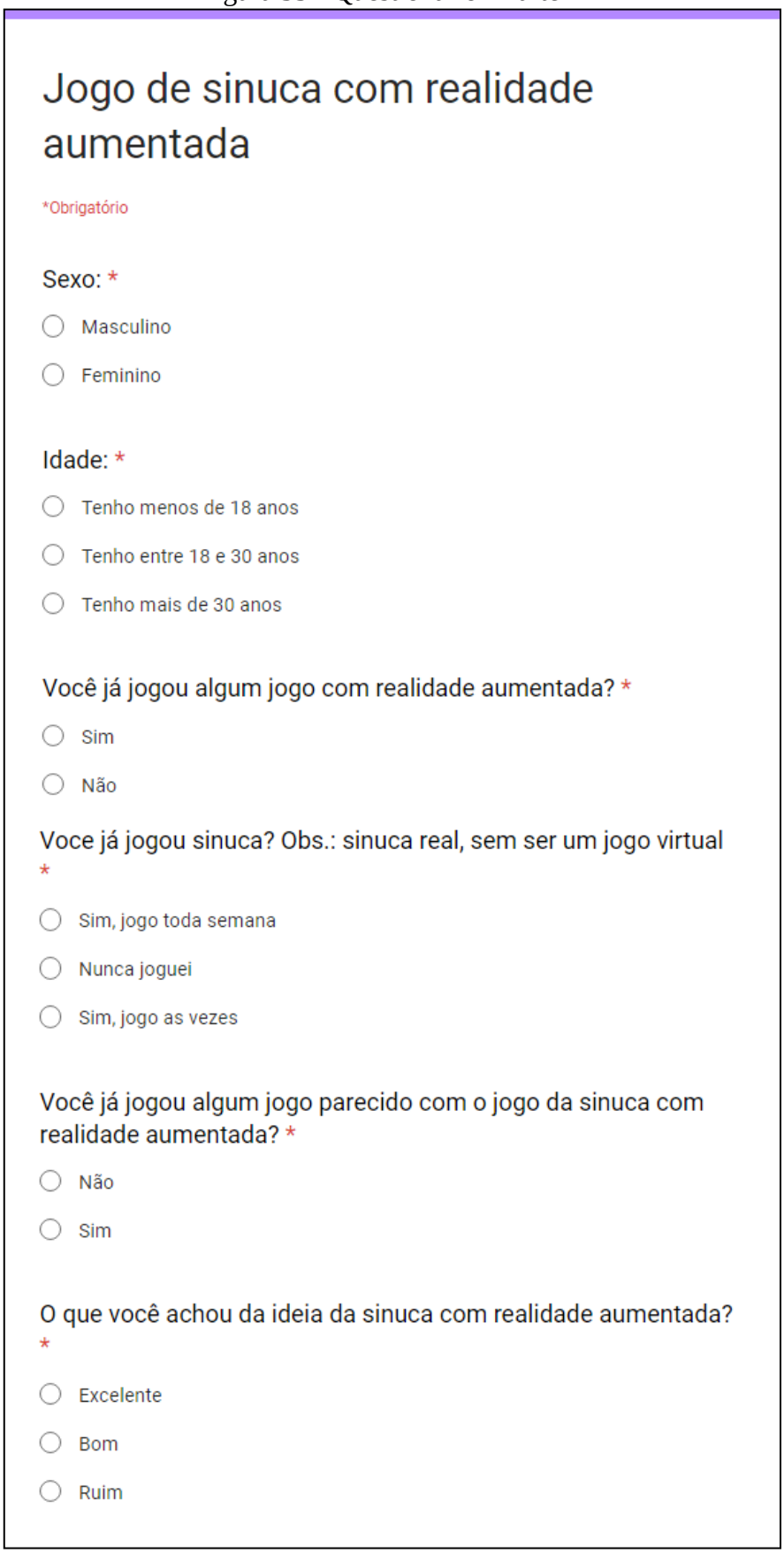
VIDA, Emerson R. G. et al. **Virtual Snooker**. 2010. 77 f. Monografia (Bacharel em Ciência da Computação) - Centro Universitário de FEI, São Bernardo do Campo.

VIEIRA, Beatriz N. S.; THEODORO, Camila; TRIAS, Lucas Padovani. **ARHockey:** Um Jogo em Realidade Aumentada Baseada em Projetores. 2006. 69 f. Trabalho de conclusão de curso (Bacharel em Ciência da Computação) - Centro Universitário Senac, São Paulo - SP.

APÊNDICE A – Roteiro do experimento e o questionário realizado

Neste apêndice constam o roteiro realizado no experimento com os convidados e o questionário respondido pelos participantes. Para realizar o experimento, o equipamento já estava ajustado e os usuários apenas jogaram sinuca. O primeiro o teste foi a utilização dos dois simulacros de taco, o de ferro e o de madeira, com a execução da rotina 1, que é o algoritmo `Primeiro Ponto Com Ultimo Ponto`. Depois de jogar com a rotina 1, os convidados executaram o segundo teste, que é a execução da rotina 2, com o algoritmo `Penultimo Ponto Com Ultimo Ponto` e jogando com os dois simulacros de sinuca. O questionário respondido pelos convidados foi divido em três parte e estão apresentadas nas Figura 35, Figura 36 e Figura 37.

Figura 35 - Questionário - Parte 1



Jogo de sinuca com realidade aumentada

*Obrigatório

Sexo: *

- Masculino
- Feminino

Idade: *

- Tenho menos de 18 anos
- Tenho entre 18 e 30 anos
- Tenho mais de 30 anos

Você já jogou algum jogo com realidade aumentada? *

- Sim
- Não

Voce já jogou sinuca? Obs.: sinuca real, sem ser um jogo virtual *

- Sim, jogo toda semana
- Nunca joguei
- Sim, jogo as vezes

Você já jogou algum jogo parecido com o jogo da sinuca com realidade aumentada? *

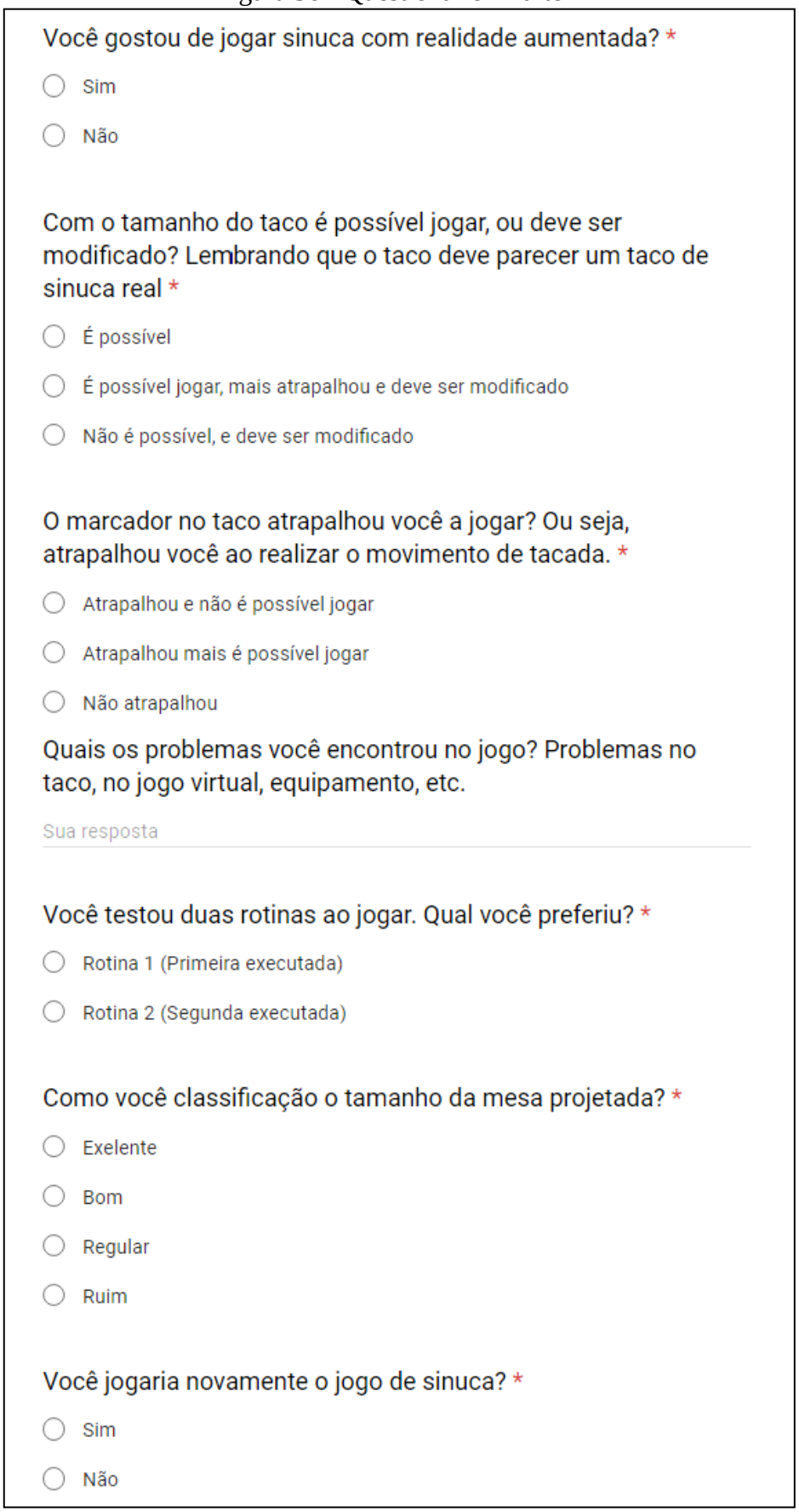
- Não
- Sim

O que você achou da ideia da sinuca com realidade aumentada? *

- Excelente
- Bom
- Ruim

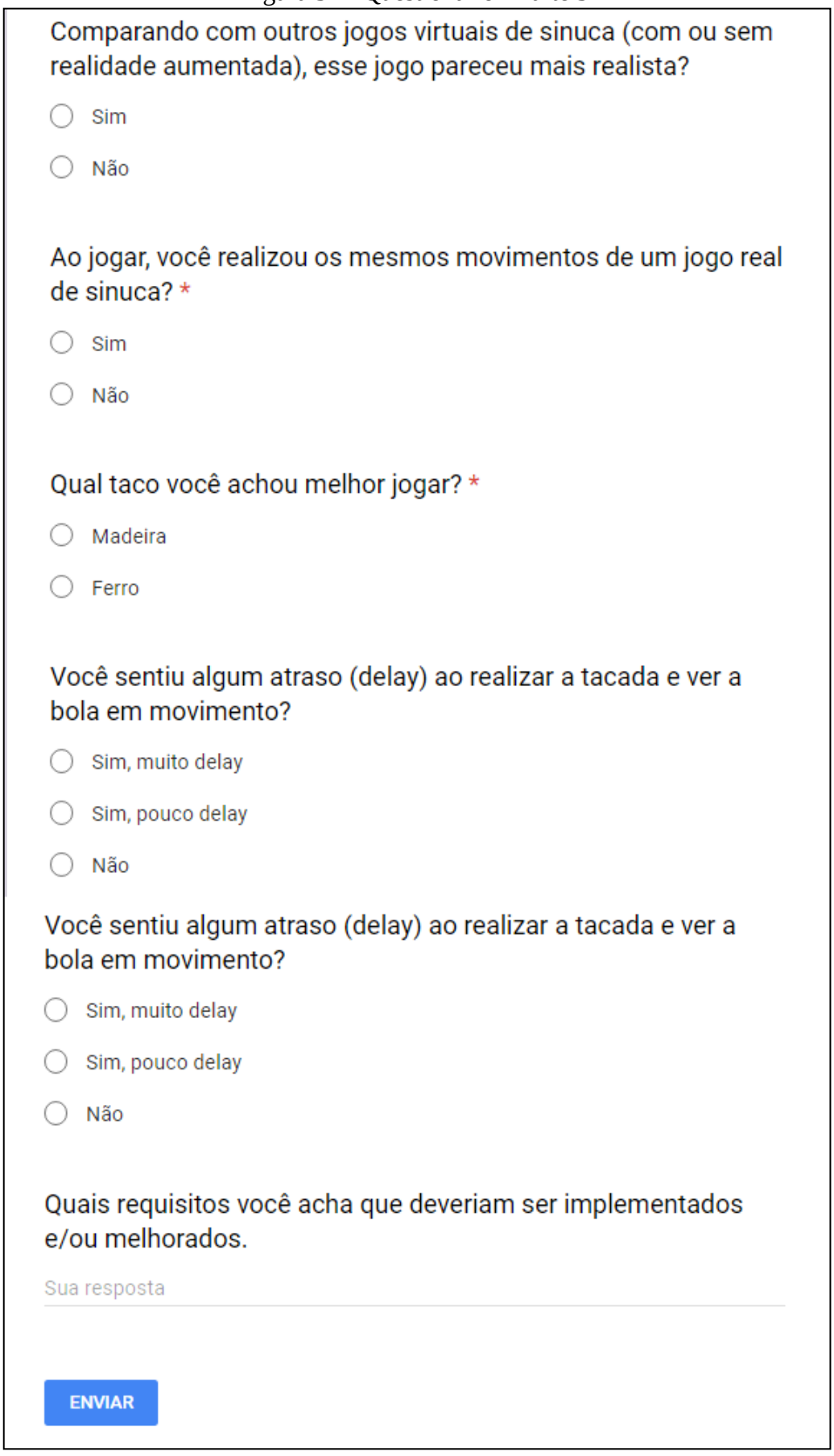
Fonte: elaborado pelo autor.

Figura 36 - Questionário - Parte 2



Fonte: elaborado pelo autor.

Figura 37 - Questionário - Parte 3



Fonte: elaborado pelo autor.