UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

PEDRO HENRIQUE ATHILA QUEIROZ DE SOUSA

INTERAÇÃO COM UM PERSONAGEM FISICAMENTE SIMULADO UTILIZANDO VISÃO COMPUTACIONAL E REALIDADE AUMENTADA

QUIXADÁ
2015

PEDRO HENRIQUE ATHILA QUEIROZ DE SOUSA

INTERAÇÃO COM UM PERSONAGEM FISICAMENTE SIMULADO UTILIZANDO VISÃO COMPUTACIONAL E REALIDADE AUMENTADA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: Computação

Orientador Prof. Dr. Rubens Fernandes Nunes

QUIXADÁ
2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S696i Sousa, Pedro Henrique Athila Queiroz de
Interação com um personagem fisicamente simulado usando visão computacional e realidade aumentada / Pedro Henrique Athila Queiroz de Sousa. – 2015.
46 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Bacharelado em Sistemas de Informação, Quixadá, 2015.
Orientação: Prof. Dr. Rubens Fernandes Nunes
Área de concentração: Computação

1. Interação homem-máquina 2. Visão por computador 3. Realidade virtual I. Título.

CDD 004.019

PEDRO HENRIQUE ATHILA QUEIROZ DE SOUSA

INTERAÇÃO COM UM PERSONAGEM FISICAMENTE SIMULADO UTILIZANDO VISÃO COMPUTACIONAL E REALIDADE AUMENTADA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: Computação

Aprovado em: _____ / Junho / 2015.

BANCA EXAMINADORA

_____
Prof. Dr. Rubens Fernandes Nunes (Orientador)
Universidade Federal do Ceará-UFC

_____
Prof. Dr. Críston Pereira de Souza
Universidade Federal do Ceará-UFC

_____
Prof[a]. Dra. Paulyne Matthews Jucá
Universidade Federal do Ceará-UFC

Aos meus amigos e família

AGRADECIMENTOS

Agradecer primeiramente a Deus, sem ele nada seria possível. A minha mãe e a meu irmão por estarem sempre presentes. Pela paciência do meu orientador Rubens Fernandes Nunes que me ajudou bastante na criação desse projeto, fornecendo muitas ideias, apoio e nunca deixando de me incentivar. A todos os professores que tive durante essa jornada em especial Wladimir Araújo Tavares, Enyo José Tavares Gonçalves, Jefferson de Carvalho Silva e Ricardo Reis Pereira. Aos meus amigos Luiz, Vanessa, Bitu, Frakon, Akirah, Catatu Nervoso, Eirien, Emilialmeida, Nimeria, SkyWish, TrusD, MissGrey, Dig SkepyStar, a todos os outros não citados que de alguma forma me deram apoio.

"Descobri o que importava antes
mesmo de descobrir o que queria."

RESUMO

Com o avanço da tecnologia, interagir com o mundo virtual está se tornando cada vez mais presente no dia a dia, seja para realizar uma ligação usando um *smartphone* ou para jogar usando o próprio corpo com o auxilio do *Microsoft Kinect*. Diante dessa realidade, buscar formas de interação com o mundo virtual visando a redução de custos torna-se uma necessidade. Desenvolvemos protótipos de interação, usando realidade aumentada e visão computacional. Para aumentar o realismo, fazemos uso de um personagem fisicamente simulado. Como forma de medir essa interação, realizamos testes com usuários para identificar o desempenho e a imersão proporcionada por cada protótipo de interação.

Palavras chave: Interação. Visão computacional. Realidade aumentada.

ABSTRACT

With the advancement of technology, interact with the virtual world is becoming increasingly present in everyday life, or to make a connection using a smartphone or play using the own body with the help of Microsoft Kinect. Given this reality, seek ways of interaction with the virtual world aiming at reducing costs becomes a necessity. We develop interaction prototypes using computer vision and augmented reality. To increase the realism, we make use of a physically simulated character. As a form to measure this interaction, we performed user testing to identify performance and immersion provided by each prototype interaction.

Palavras chave: Interaction. Computer vision. Augmented reality.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O uso de interfaces com o usuário, que o permitem interagir com mundos virtuais, está se tornando cada vez mais comum em nossa vida diária, como evidenciado pela alta demanda por *Microsoft Kinect* desde o seu lançamento em 2010. A qualidade da experiência sentida pelos participantes depende muitas vezes da forma com que o ambiente e os personagens virtuais demonstram interação.

No contexto de um jogo, existem diferentes formas de interação entre um usuário real (jogador) e o mundo virtual (jogo), as quais são dadas através do uso de efeitos visuais, sons, animações etc. Ao controlar um personagem virtual através de algum dispositivo de entrada, o jogador exerce uma determinada ação no jogo. O jogo, por sua vez, executa efeitos como forma de resposta, interagindo com o jogador.

Uma possibilidade de interação com o mundo virtual é usar um equipamento de captura de movimentos em tempo real a fim de simular a interação direta entre uma pessoa real e o ambiente virtual. No caso, o ator executa a ação e seu avatar imita sua atuação praticamente ao mesmo tempo em que ela ocorre, dando vida ao personagem animado.

Segundo Liu e Zordan (2011), integrar o desempenho do avatar em tempo real com um ambiente fisicamente simulado apresenta dois grandes desafios. Em primeiro lugar, devido a inconsistências entre o mundo virtual e o real, os movimentos que o ator faz para executar uma determinada ação no mundo real podem se tornar artificiais quando usados diretamente no ambiente virtual. Liu e Zordan (2011) citam um exemplo no qual o ator está em pé e tenta controlar o avatar para fazer uma travessia, suspenso pelos braços, por uma sequência de barras horizontais (*monkeybar*). O movimento que o ator realiza é inconsistente com o movimento do avatar, pois enquanto o ator está apoiado no chão e apenas movimentando os braços para simular o percurso, o avatar deve apresentar um movimento natural de balanço, consequente do contato entre as mãos e a barra, e da gravidade agindo no personagem suspenso. O segundo grande desafio, mais relacionado a este trabalho, está no fato de que, se existe interação física entre o avatar e um objeto virtual (ou outro personagem), tanto o personagem quanto o objeto devem responder ao impacto da interação. Empregar simulação física é geralmente mais adequado para reproduzir os movimentos alterados por consequência desses impactos

De maneira geral, simulação física tem sido bastante utilizada devido à grande capacidade de criar movimentos realistas de maneira automática (KIM; KIM; LEE, 2011).

Neste trabalho, propõe-se comparar diferentes formas de interação com um personagem fisicamente simulado. Em vez de usar equipamentos caros de interação, como é o caso de equipamentos ópticos de captura de movimentos, ou até mesmo do próprio *Kinect*, o qual já vem se difundindo em uma proporção bem maior, propõe-se analisar o uso de uma simples *webcam*, apenas combinada com o uso de técnicas de visão computacional e realidade aumentada. Além de serem de mais difícil acesso, esses equipamentos mais caros e sofisticados também possuem a desvantagem de exigir instalação, configuração e utilização mais complicadas. Por outro lado, câmeras de vídeo comuns já estão bastante difundidas e acessíveis.

A fim de tornar essa interação o mais natural possível, o personagem é projetado no mundo real com o uso de realidade aumentada, e a visão computacional é usada para identificar os movimentos do usuário. Pretendemos, assim, aumentar a sensação de imersão do usuário no ambiente virtual. Em outra perspectiva, queremos causar a sensação de que o personagem realmente existe no mundo real. Enquanto o usuário vê e interage com o personagem, o personagem reage às ações do usuário.

Dentro desse contexto, o objetivo geral deste trabalho consiste em implementar e avaliar a interação com um personagem fisicamente simulado usando técnicas de visão computacional, com o uso da biblioteca *OpenCV*, e realidade aumentada, com o uso da biblioteca *ARToolKit*. Propõe-se criar protótipos de interação com o personagem fisicamente simulado, comparar o uso das bibliotecas ARToolKit e OpenCV, e analisar a imersão proporcionada pelo fato de se estar interagindo com personagens fisicamente simulados.

2 FUNDAMENTAÇÃO TEÓRICA

Para a realização deste trabalho, usamos técnicas de realidade aumentada para projetar objetos virtuais no mundo real. Para desenhar esses objetos, foi usada a ferramenta OpenGL. Para identificar um objeto real e reproduzir suas coordenadas em um objeto 3D, foi utilizada visão computacional. O uso de simulação física tornou-se necessário para dar a esses objetos virtuais um sentimento de realismo, adicionando elementos físicos como gravidade, força etc.

2.1 Visão computacional

Visão computacional é a ciência responsável pela visão de uma máquina, ou seja, pela forma com que o computador consegue enxergar o ambiente a sua volta, extraindo informações capturadas por câmeras de vídeo, *scanners*, entre outros dispositivos de captura de imagens. Segundo Azevedo et al. (2008), visão computacional combina a área de análise de imagens com técnicas de inteligência artificial, dando ao computador a capacidade de ver, identificar objetos, interpretar o ambiente a sua volta, extrair informações dele e executar ações.

Segundo Azevedo et al. (2008), sistemas que usam visão computacional vêm sendo usados em reconhecimento de pessoas, de assinaturas e de objetos; inspeção de peças em linhas de montagem; orientação de movimentos de robôs em indústrias automatizadas etc. Como um exemplo desses sistemas, Bianchi e Reali-costa (2000) usam visão computacional para controlar um time de futebol de robôs. Uma câmera posicionada sobre o campo de futebol é conectada ao computador, permitindo que ele tenha uma visão geral de todos os elementos presentes. Assim, o computador é capaz de identificar tanto o posicionamento dos robôs quanto a posição da bola no campo, permitindo que estratégias de movimentação dos robôs sejam elaboradas de acordo com essas informações.

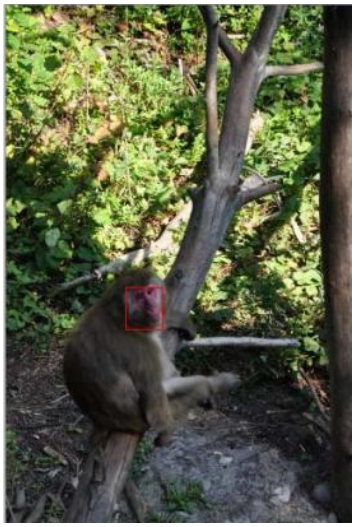
2.1.1 **Open source computer vision (OpenCV)**

OpenCV é uma biblioteca de visão computacional de código aberto escrita em C e C++, com suporte para *Windows*, *Linux*, *Android* e *Mac OS* (BRADSKI; KAEHLER, 2008), e contém mais de 2500 algoritmos otimizados (OpenCV, 2013, tradução nossa). Como alguns exemplos da capacidade desses algoritmos, a biblioteca OpenCV pode ser usada para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, acompanhar os

movimentos de câmera, rastrear objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos 3D a partir de câmeras estéreo, encontrar imagens similares pertencentes a um determinado banco de imagens, remover olhos vermelhos de imagens tiradas com flash, rastrear movimentos de olhos etc.

Junto com diferentes dispositivos de captura de imagens, essa biblioteca é usada em diversas aplicações em vários lugares do mundo. Como alguns exemplos de aplicações atualmente funcionando, ela é usada para detectar invasores em câmeras de vigilância em Israel, detectar acidentes de afogamento em piscinas na Europa, ajudar robôs a encontrar e pegar objetos em um famoso laboratório de pesquisa em robótica (*Willow Garage)*, e inspecionar rótulos de produtos nas fábricas ao redor do mundo (OpenCV, 2013, tradução nossa). A Figura 1 ilustra o uso do OpenCV no reconhecimento facial. Nessa figura, o sistema consegue detectar o rosto de um animal, destacado com um retângulo vermelho.

Figura 1 – Exemplo de uso do OpenCV



Fonte: Laganière, (2011)

O OpenCV pode ser explorado em diversas áreas da tecnologia, inclusive combinado com outras bibliotecas, como a biblioteca gráfica OpenGL, integrando visão computacional e computação gráfica (síntese de imagens).

2.2 Realidade aumentada

Realidade Aumentada (RA) permite ao usuário visualizar o mundo real com objetos virtuais sobrepostos. Essa combinação entre real e virtual tem como objetivo trazer para o ambiente real as facilidades existentes no mundo imaginário, produzido por

computadores. Uma das vantagens é que objetos virtuais podem ser facilmente manipulados sem as restrições da física. Por exemplo, passar alguns comandos para o computador é suficiente para posicionar um guarda-roupa em um determinado local de uma sala, independente de quão pesado ele possa parecer. Por outro lado, manter algumas características físicas é importante para que os componentes virtuais tenham uma maior aceitação por parte do usuário, em alguns casos podendo causar a sensação de que essas entidades virtuais de fato fazem parte do mundo real. A maneira como a interação com tais entidades virtuais é tratada pode ser considerada como um fator determinante para essa aceitação.

Para que essa interação seja possível, um dispositivo de captura de imagens, funcionando em tempo real, é necessário (KIRNER; SISCOUTTO, 2007). As imagens capturadas são analisadas e técnicas de visão computacional são utilizadas para detectar movimentos específicos que podem ser usados para controlar objetos virtuais, através da atualização de suas posições e orientações. Marcadores são geralmente utilizados para ajudar na detecção desses movimentos, facilitando a manipulação das informações virtuais. Esse mapeamento de movimentos reais em movimentos virtuais torna a interação do usuário com o mundo virtual mais natural. Por exemplo, o usuário pode usar suas próprias mãos para interagir com objetos virtuais mostrados na cena.

Motivados por essa interação natural, diversos sistemas de RA vêm sendo desenvolvidos. Na indústria de jogos, por exemplo, o console *Playstation Vita* possui funções de RA, permitindo que o ambiente real seja utilizado como o ambiente do jogo.

2.2.1 **Sistemas de realidade aumentada**

Através dos sistemas de realidade aumentada, o usuário é capaz de se comunicar mais facilmente com o mundo virtual devido a essa troca de informações visuais, realizada em um contexto familiar. Isso permite que a aprendizagem da utilização desses sistemas seja adquirida naturalmente, sem a exigência de um extenso treinamento ou adaptação (TORI; KIRNER; SISCOUTTO, 2006).

Figura 2 – Jogo PulzAR

Fonte: PulzAR™, (2013)

Um exemplo de um sistema desse tipo é o jogo PulzAR (MONTEIRO, 2013), disponível para o console *Playstation Vita* (Figura 2). O objetivo do jogo é proteger a terra de um grande asteroide que está vindo destruí-la. Os marcadores de realidade aumentada são usados para representar os objetos do jogo e cada objeto possui sua função específica. O jogador tem que movimentar os marcadores para resolver um enigma. Quando solucionado, um foguete é ativado e destrói o asteroide. Para analisar melhor o enigma, o sistema também permite que o jogador visualize o quebra-cabeça de diferentes ângulos. O controle do ângulo de visão é realizado através da simples movimentação de um marcador específico.

2.2.2 **ARToolKit**

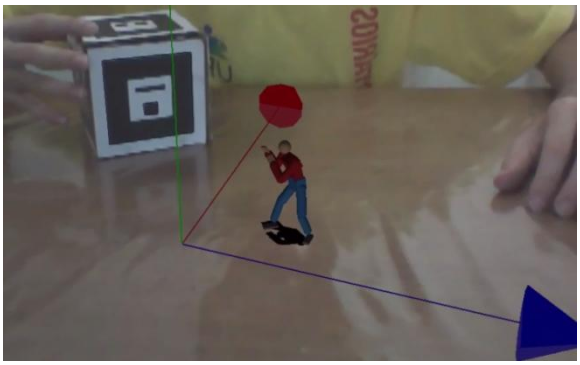
O ARToolKit é uma biblioteca de programação multi-plataforma, considerada um kit de ferramenta de Realidade Aumentada (RA), sendo bastante utilizada e discutida por desenvolvedores e pesquisadores da comunidade de RA (SANTIN; KIRNER, 2008). O ARToolKit possui o seu código livre para modificações e uso no desenvolvimento de aplicações não comerciais sob licença GPL.

A biblioteca ARToolKit é implementada em C e C++ oferecendo suporte a programadores para o desenvolvimento de sistemas de RA. Segundo Santin e Kirner (2008), a biblioteca utiliza rastreamento óptico, que implementa técnicas de visão computacional para identificar e estimar em tempo real a posição e a orientação de um marcador em relação ao dispositivo de captura de imagens. Aplicações desenvolvidas com ARToolKit podem ser vistas através de dispositivos de visão indireta, não imersivos, ou visão direta, imersivos. Por exemplo, um monitor pode ser considerado um dispositivo não imersivo, pois o usuário não necessariamente olha na direção da cena real gravada pela câmera. Já usando óculos virtuais com câmeras acopladas, o usuário tem a cena real acontecendo na sua frente. Quanto mais

imersivo for o dispositivo de visão, maior será a sensação do usuário de que esses objetos realmente pertencem ao ambiente real.

Aplicações desenvolvidas com ARToolKit podem ser implementadas com diferentes bibliotecas gráficas, tais como OpenGL e VRML. Neste trabalho, usamos a biblioteca OpenGL. A visualização dos objetos virtuais é realizada no momento da inserção de seus respectivos marcadores no campo de visão do dispositivo de captura de imagens (SANTIN; KIRNER, 2008). Para um melhor entendimento, a Figura 3 mostra um boneco desenhado relativo ao seu marcador.

Figura 3 – Personagem com seu marcador.



Fonte: Criada pelo autor, (2015)

2.2.3 **Marcadores**

Marcadores reconhecidos pelo ARToolKit são figuras geométricas quadradas com símbolos de identificação em seu interior (SANTIN; KIRNER, 2008). A biblioteca vem com alguns marcadores pré-fabricados e um arquivo *Blankpattern.png* para a criação de novos marcadores (ARToolWorks, 2015, tradução nossa).

O rastreamento que está implementado no ARToolKit é capaz de estimar a posição e a orientação dos marcadores, informações muito úteis para o desenvolvimento de aplicações. Essas informações são suficientes para definir uma configuração única para qualquer objeto rígido tridimensional ou, de maneira mais geral, qualquer sistema de coordenadas tridimensional. A obtenção dessas informações é realizada através da análise da imagem captada pelo dispositivo de vídeo, utilizando visão computacional, que estabelece o relacionamento entre as coordenadas do marcador e as coordenadas da câmera (SANTIN; KIRNER, 2008).

O uso de marcadores possibilita uma interação mais direta entre o usuário e os objetos virtuais, os quais são desenhados de acordo com os marcadores. Ou seja, na medida em que o usuário movimenta um marcador, o objeto virtual é atualizado em tempo real.

2.2.4 **Criação de marcadores**

O ARToolKit permite a criação de novos marcadores. Essa criação é dividida em duas partes. A primeira parte é a criação do arquivo de imagem do marcador, que deve seguir algumas restrições (ARToolWorks, 2015, tradução nossa):

- Devem ser quadrados.
- Devem ter uma borda contínua (geralmente preto ou branco puro) e a área de dentro da borda deve possuir uma cor de fundo contrastante (geralmente o oposto da cor da borda). Por definição, a espessura da borda é 25% do comprimento do marcador.
- A figura de dentro da borda, a qual nos referimos como a imagem do marcador, não deve ser simétrica de rotação (ou seja, a imagem do marcador não dever ser a mesma ao realizar rotações de 90º, 180º e 270º no marcador). A imagem pode ser preta, branca ou colorida.

A Figura 4 é um exemplo de um marcador que pode ser usado no ARToolKit.

Figura 4 – Marcador do ARToolKit.

Fonte: Santin; Kirner, (2008)

A segunda parte é o treinamento do arquivo de imagem do marcador. Esse treinamento gera o arquivo *Pattern*, que é o arquivo que contém dados que representa a

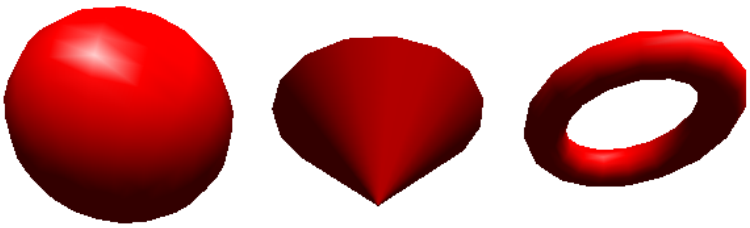
imagem no centro de um marcador. Essa parte pode ser feita usando o próprio ARToolKit ou através da ferramenta ARToolKit Marker Generator (ARToolWorks, 2015, tradução nossa).

2.2.5 **OpenGL**

OpenGL é uma API (*Application Programming Interface*) para o desenvolvimento de aplicações de computação gráfica com suporte a *Linux*, *Windows* e *Mac OS*, podendo ser usada com as linguagens de programação Ada, C, C++, Fortran, Python, Perl e Java, e oferece total independência de protocolos de rede e topologias (OpenGL, 2014, tradução nossa).

Possuindo uma rotina simples para o desenvolvimento de softwares gráficos, OpenGL fornece aos desenvolvedores de software acesso às geometrias, listas de exibição, transformações de modelagem, texturização, iluminação, *anti-aliasing* e muitas outras características (OpenGL, 2014, tradução nossa). Desde sua criação em 1992, o OpenGL tornou-se a API mais utilizada para a criação de aplicativos 2D e 3D na indústria de computação gráfica (OpenGL, 2014, tradução nossa). A Figura 5 mostra o uso da biblioteca na criação de figuras geométricas em 3D.

Figura 5 – Figuras geométricas
criadas usando o OpenGL.



Fonte: Criada pelo autor, (2015)

2.3 Interação

Segundo Leffa (2013), interação é um processo que envolve dois ou mais elementos, sejam eles partículas, corpos ou pessoas, não existindo interação de elemento único. Na sua essência, interação parte da ideia de contato, podendo ser definida como um contato que produz mudança em cada um dos participantes.

Uma interação pode ocorrer entre seres de mesma natureza ou de natureza diferentes, como por exemplo, entre pessoas e animais. De modo geral, uma interação ocorre quando uma entidade exerce uma ação e essa ação provoca a reação de uma ou mais entidades, seguindo a lei da física que toda ação provoca uma reação. Por exemplo, podemos citar a interação entre um usuário e seu *smartphone,* em que o usuário toca no aplicativo que deseja abrir. O *smartphone* por sua vez produz alguma animação para dar a sensação de toque. Essa animação é a maneira pela qual o *smartphone* responde a interação do usuário.

A interação entre objetos reais e virtuais é uma das principais preocupações na pesquisa de realidade aumentada. Quando simulação física é usada para reproduzir os movimentos, no momento em que um objeto virtual colide com um objeto real, ambos os objetos devem ser afetados pela colisão. Porém, só é possível controlar o movimento do objeto virtual. O objeto real permanece inalterado. Essa interação unidirecional diminui a sensação de realismo (KIM; KIM; LEE, 2011).

2.4 Simulação física de personagens virtuais

Enquanto animações que não se baseiam em física dependem do talento artístico dos animadores, o uso de simulação física garante automaticamente que o movimento resultante obedecerá às leis da física, consistindo em uma abordagem bastante promissora para a animação por computador. Por um lado, isso facilita o trabalho do animador, pois ele não precisa se preocupar com a corretude física do movimento. Por outro lado, como todo o movimento é resultante da aplicação de forças e toques, e não da manipulação direta das posições dos objetos, o controle da animação é dificultado. Esperamos que novas formas de interação possam facilitar o controle dessas animações se adequadamente incorporadas.

Note que o foco deste trabalho é analisar o quanto o uso de personagens fisicamente simulados pode contribuir em relação à sensação de imersão por parte do usuário. A implementação da parte de simulação física propriamente dita não faz parte do escopo deste trabalho. O código do personagem simulado foi fornecido por Nunes et al. (2008), e por isso escolhido para ser usado. Caso o leitor queira se aprofundar no assunto deste tópico, sugerimos também a leitura do trabalho de Geijtenbeek e Pronost (2012).

2.4.1 **Open dynamics engine (ODE)**

Open Dynamics Engine (ODE) é uma biblioteca de alto desempenho para simulação dinâmica de corpos, independente de plataforma e fácil de usar com C/C++. Possui

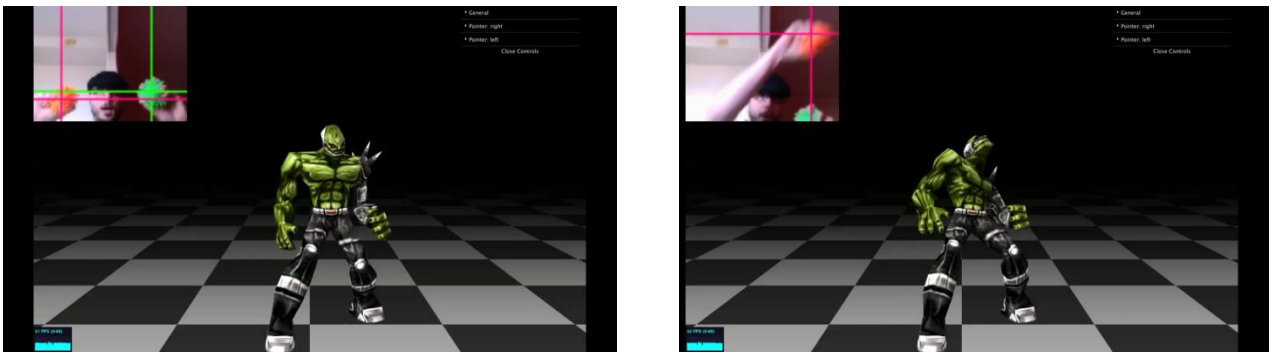
funções avançadas de detecção de colisão e permite tratar atrito. Sendo uma biblioteca útil para simular veículos, objetos em ambientes de realidade virtual e criaturas virtuais, é usada atualmente em muitos jogos de computador, ferramentas de criação 3D e ferramentas de simulação (ODE, 2014, tradução nossa).

2.5 Trabalhos relacionados

Nesta seção, dois trabalhos que serviram de base para a criação do projeto são apresentados, destacando suas semelhanças e diferenças.

Etienne (2014) mostra que é possível interagir com um personagem virtual movimentando bidimensionalmente pequenas bolas no mundo real. Esses movimentos são mapeados em socos no personagem (Figura 6). Porém, o impacto da interação com o personagem é pré-determinado, consistindo apenas em algumas animações fixas que são escolhidas de acordo com o movimento da bola realizado pelo usuário. O sistema proposto fornece ao usuário a sensação de socar o personagem apenas de duas maneiras diferentes. Além disso, nenhuma análise de profundidade é feita no movimento real do usuário. No nosso caso, usamos um personagem fisicamente simulado, em que cada interação diferente do usuário resulta em uma animação diferente do personagem.

Figura 6 – Exemplo de interação
com personagem.



Fonte: Etienne, (2014)

Song et al. (2008) utilizam simulação física e uma *webcam* para criar um jogo de pesca (Figura 7). Nesse jogo, o dedo do jogador torna-se uma vara de pescar quando ele está no campo de visão da *webcam*. Em outras palavras, a vara de pescar segue a trajetória definida pelo movimento do dedo do jogador, enquanto elementos físicos, tais como a gravidade e outras forças externas, atuam sobre ela, dando a sensação de um movimento real. Todas as interações que ocorrem no jogo são influenciadas pela simulação física, aumentando

o realismo. Nesse trabalho, vemos o uso de visão computacional para controlar um objeto virtual e a adição de física para controlar o impacto entre os objetos. Diferente desse trabalho, nós não pretendemos criar um cenário de jogo. Nossa proposta é a criação de um protótipo em que podemos interagir com um personagem virtual, controlando objetos virtuais através do mapeamento de objetos reais.

Figura 7 – Jogo de pesca.

Fonte: Song et al. (2008)

Nesses trabalhos, podemos ver a interação com o mundo virtual sendo feita através de algum objeto ou até mesmo usando partes do próprio corpo. Esses recursos que permitem interagir com o mundo virtual estão sendo bastante usados na criação de jogos, tais como o jogo PulzAR (MONTEIRO, 2013) e outros diversos jogos que usam *Microsoft Kinect. O* fato do usuário usar o próprio corpo ou um simples objeto para interagir pode aumentar a sensação de imersão do usuário no ambiente virtual, descartando a necessidade de treinamento e possibilitando que o usuário aprenda os comandos de forma natural.

3 PROCEDIMENTOS METODOLÓGICOS

Este projeto de pesquisa consiste em analisar a influência do uso de personagens fisicamente simulados na interação por visão computacional e realidade aumentada. Para isso, criamos protótipos de interação, usando OpenCV e ARToolKit, para analisar essa influência em diferentes situações. Enquanto o ARToolkit já é integrado com o OpenGL, a integração entre o OpenCV e o OpenGL precisou ser implementada. Como já mencionado, a implementação do personagem fisicamente simulado não faz parte do escopo. Nunes et al. (2008) disponibilizaram o código para a realização deste trabalho. A biblioteca ODE é utilizada para realizar a simulação física e o tratamento de colisão entre os objetos.

Usando algoritmos de rastreamento do OpenCV para identificar um objeto no mundo real, é possível fazer uma associação entre as coordenadas reais desse objeto identificado e as coordenadas de um objeto virtual, o qual é continuamente atualizado. Uma vez integrados (OpenCV e OpenGL), esse objeto virtual pode ser mostrado para o usuário, em tempo real, e é usado para interagir com o personagem fisicamente simulado.

Com o ARToolKit, marcadores independentes são usados para definir os sistemas de coordenadas do personagem e do objeto usado para interagir com ele. Ambos são desenhados sobre o mundo real.

3.1 Protótipo openCV

Para que o objeto real seja bem identificado pelos algoritmos disponíveis pelo OpenCV, é recomendável que ele possua uma cor uniforme e destacada do resto do ambiente. Com o intuito de facilitar os primeiros testes, uma luva de cor laranja foi usada como o objeto real a ser rastreado pela câmera. Usando técnicas de visão computacional, o objeto real foi rastreado e suas coordenadas foram mapeadas para as coordenadas de um objeto virtual. Para simplificar o mapeamento, a orientação 3D do objeto rastreado é ignorada, e apenas a sua posição 3D é considerada. Assim, uma esfera é suficiente para representar o objeto virtual, o qual é desenhado usando o OpenGL. Essa esfera é desenhada no mesmo ambiente 3D em que o personagem virtual fisicamente simulado se encontra. Depois de vários testes, a luva foi substituída por uma bola de frescobol avermelhada, devido à maior similaridade com o objeto virtual utilizado. O monitor do computador é usado para visualizar o ambiente 3D, enquanto

uma simples câmera, conectada ao computador, é usada para detectar o movimento do objeto real.

3.1.1 **Integração openCV com openGL**

Integramos o OpenCV com o OpenGL usando o algoritmo de rastreamento *CamShift* do OpenCV para identificar um objeto do mundo real. As coordenadas desse objeto real são mapeadas para as coordenadas de uma esfera virtual. Assim, todos os movimentos realizados no objeto são reproduzidos na esfera. Contudo, o algoritmo *CamShift* não identifica profundidade diretamente. A solução proposta é definir a profundidade do objeto real baseada na dimensão de uma elipse que envolve o objeto rastreado, fornecida pelo próprio OpenCV. Quando a elipse aumenta de tamanho, significa que o objeto real está se aproximando da *webcam*, pois está tomando uma maior parte do campo de visão dela. Já quando a elipse diminui, significa que ele está se distanciando dela. Com isso, conseguimos reproduzir a movimentação da coordenada Z da esfera mapeada. Com essa associação entre o tamanho da elipse e a profundidade da esfera, anexamos o personagem fisicamente simulado ao mesmo espaço virtual da esfera permitindo a interação com o personagem no mundo virtual, através de um objeto real.

3.1.2 **Mapeamento da posição 3D de uma esfera usando openCV**

Usamos o algoritmo de rastreamento *CamShift* para rastrear um objeto do mundo real. Esse algoritmo faz o rastreamento baseado na cor desse objeto. Uma das entradas do algoritmo *CamShift* é uma imagem desse objeto que deseja ser rastreado, a qual é usada em uma fase de treinamento para que sua cor seja identificada. A saída consiste em uma elipse (ou um retângulo) envolvendo o objeto rastreado, a qual é continuamente atualizada e pode ser desenhada sobre o objeto, em tempo real. Note que o tamanho da elipse depende da distância do objeto para a *webcam*. A Figura 8 ilustra o funcionamento do algoritmo. A desvantagem desse algoritmo é que ele necessita de um ambiente adequado para realizar o rastreamento. Como ele faz o rastreamento baseado na cor do objeto, ele pode se confundir com outras cores parecidas presentes no ambiente. Por exemplo, se estamos rastreando um objeto de cor vermelha e o ambiente onde esse objeto está presente também é vermelho, o algoritmo não conseguirá identificar bem o objeto no ambiente. Entretanto, com ajustes

adequados nos parâmetros (Matiz, Saturação e Brilho), o algoritmo se mostrou bastante eficaz e preciso nos testes realizados com a bola de frescobol.

Figura 8 – Algoritmo *CamShift* em funcionamento.

Fonte: Criada pelo autor, (2014)

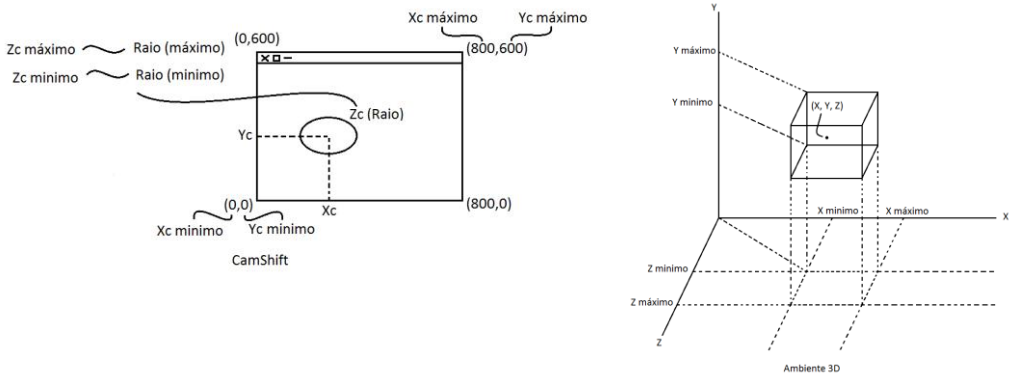
Na Figura 8, a fase de treinamento é ilustrada na primeira foto. Como foi dito, a entrada do algoritmo é uma imagem. Nesse caso, a imagem que é passada está dentro do retângulo vermelho. As demais fotos mostram o algoritmo funcionando. Uma vez identificada a cor do objeto na fase de treinamento, o algoritmo entende que o objeto é composto por todos os pixels na vizinhança que possuem uma cor próxima.

Para realizar um mapeamento tridimensional do objeto real para um objeto virtual é necessário converter as coordenadas do objeto real para as coordenadas correspondentes do objeto virtual. As coordenadas do objeto real são obtidas a partir da elipse que o envolve. Entretanto, embora a elipse produzida pelo *CamShift* seja bidimensional, três coordenadas (Xc,Yc,Zc) são necessárias para que o mapeamento seja realizado. Como mapear as informações bidimensionais da janela do OpenCV para as coordenadas tridimensionais da esfera no mundo virtual do OpenGL? Uma solução é obter Xc e Yc a partir do centro da elipse, e a coordenada Zc a partir do tamanho da elipse, mais especificamente a partir do menor raio da elipse.

Usamos o OpenGL para criar um ambiente 3D que será o espaço de atuação da esfera. Nesse mesmo ambiente, o personagem fisicamente simulado é inserido, permitindo que ocorra a interação entre o personagem e a esfera. Para movimentar a esfera, associamos as coordenadas (Xc, Yc, Zc) do *CamShift* com as coordenadas (X, Y, Z) da esfera. Note que as coordenadas (Xc, Yc, Zc) são relativas à janela da *webcam* e suas dimensões podem ser diferentes do ambiente 3D. Assim, não podemos fazer uma associação direta de (Xc, Yc, Zc) com (X, Y, Z). Criamos uma função que determina para cada coordenada (Xc, Yc, Zc) o seu respectivo valor em (X, Y, Z). Usamos funções paramétricas baseadas nos valores mínimo e

máximo de cada coordenada, tanto relativos ao sistema de coordenadas da janela quanto relativos ao sistema de coordenadas do mundo. . Sendo $Xc = Xc_{min} + u * (Xc_{max} - Xc_{min})$ a função para qualquer valor de Xc e $X = X_{min} + u * (X_{max} - X_{min})$ a função que representa qualquer valor de X no ambiente 3D, então $X = X_{min} + \frac{Xc - Xc_{min}}{Xc_{max} - Xc_{min}} * (X_{max} - X_{min})$ é a função de mapeamento do respectivo valor de Xc em X. Note que o parâmetro u é o mesmo nas duas funções. Valores de u entre 0 e 1 correspondem a valores de Xc e X entre seus respectivos valores mínimos e máximos. Essa mesma regra pode ser aplicada para Yc e Zc, obtendo assim todas as coordenadas necessárias para realizar um mapeamento 3D. A figura 9 ilustra esse mapeamento.

Figura 9 – Mapeamento 3D



Fonte: Criada pelo autor, (2014)

3.2 Protótipo ARToolkit

O protótipo com o ARToolKit tem duas versões. Em uma delas, a interação é feita com personagens em tamanho reduzido (miniatura). Na outra, o personagem possui o tamanho de uma pessoa normal.

Nas duas versões, uma caixa com cinco marcadores, um em cada face, é usada para realizar o posicionamento inicial e a interação com o personagem. O uso dessa caixa foi motivada pelo aplicativo ARis (ARis, 2015).

No protótipo de tamanho reduzido, apenas um dos marcadores da caixa representa o sistema de coordenadas do personagem. Depois de apertar um determinado botão, esse sistema coordenadas torna-se fixo em relação à câmera e a caixa passa a ser mapeada no objeto virtual, permitindo a interação através da sua movimentação em direção ao personagem.

O protótipo de tamanho real usa a mesma caixa, porém não restringe o uso de um marcador específico para representar o sistema de coordenadas do personagem. Quando apertado um determinado botão, o sistema de coordenadas representado pelo marcador mais visível no momento é utilizado e fixado.

A visualização é feita com o uso de um computador e de uma *webcam* conectada a ele. A *webcam* é apontada para os marcadores e as imagens dos objetos podem ser visualizadas no computador.

3.2.1 **Interação com personagens em miniatura**

Este trabalho se baseia em um protótipo previamente implementado, também disponibilizado por Nunes et al. (2008), em que a interação com personagens em miniatura usando ARToolKit foi testada. Nesse protótipo anterior, enquanto um dos marcadores representava o personagem, outro marcador menor representava um cilindro. O tamanho do marcador deve ser proporcional ao tamanho do objeto mapeado. Note que a caixa com cinco marcadores não era utilizada, e seu rastreamento foi implementado nos protótipos mostrados neste trabalho para facilitar a manipulação do objeto virtual. Um personagem inicialmente em miniatura foi usado porque os primeiros testes foram feitos usando os marcadores muito próximos à câmera. Consequentemente, para que o personagem fosse mostrado por completo na tela, ele precisava ser reduzido.

Além da interação direta entre o cilindro e o personagem, esse protótipo inicial também permite que o usuário atire pequenas esferas no personagem, apontando o cilindro na direção desejada e apertando uma tecla. Note que o ARToolkit também fornece informações de orientação, não apenas informações de posição.

Figura 10 – Interação com o personagem em miniatura.

Fonte: Nunes et al, (2008)

Na Figura 10, podemos ver o personagem fisicamente simulado na primeira foto. A segunda foto ilustra a interação através de tiros em direção ao personagem. As duas outras

fotos mostram o contato direto com o personagem, usando o cilindro. Note que os marcadores usados não aparecem explicitamente na imagem porque estão escondidos atrás dos quadrados brancos.

3.2.2 Interação com personagens em tamanho real

Na intenção de tornar a interação mais natural, usamos um personagem com tamanho de aproximadamente 1,5m a 1,6m. Para alterar o tamanho do personagem, foi preciso acrescentar um fator de escala em algumas poucas partes específicas do código fonte disponibilizado por Nunes et al. (2008), responsável por dimensionar adequadamente o modelo usado para o personagem, tanto na simulação física quanto na detecção de colisão. O valor do fator pode ser alterado via teclado, permitindo ajustar o tamanho do personagem de acordo com o tamanho desejado.

Note que com esse tamanho não é mais possível posicionar a *webcam* próxima ao marcador, como foi feito no protótipo em miniatura, pois o tamanho do personagem é maior e com a *webcam* próxima só seria possível ver uma parte do personagem. Entretanto, ao posicionar os marcadores longe da câmera, ela não consegue identificá-los muito bem, devido às limitações de resolução. Para diminuir o impacto desse problema, após o marcador ser reconhecido pela *webcam,* gravamos suas informações de posicionamento e orientação, deixando-as fixas em relação à *webcam.*

Além disso, a interação deve ser realizada em um ambiente espaçoso devido ao novo tamanho do personagem. O marcador do personagem é posicionado adequadamente no chão, suas informações são gravadas (deixando-as fixas) e o mesmo marcador é usado pelo usuário para realizar a interação. Uma dificuldade percebida é posicionar adequadamente a *webcam* para que ela tenha uma boa visão do marcador.

3.3 Avaliação da interação

Testes envolvendo usuários foram realizados para identificar qual protótipo e qual biblioteca forneceu a interação mais natural.

Os usuários foram escolhidos mediante a divulgação da realização dos testes. Os usuários interagiram com todos os protótipos criados: os dois protótipos usando ARToolKit, mostrando o personagem em miniatura e em tamanho real, e o protótipo usando OpenCV. Os

usuários realizaram uma rotina previamente definida, com o objetivo de fazer o usuário perceber as diferentes formas de reação do personagem. Ao final dos testes, os usuários descreveram na forma de relatos como foi a sensação de interagir com o personagem.

Esses relatos serviram de base para se ter uma ideia de quanto o usuário se sentiu imerso e para obter *feedback* para futuras funcionalidades.

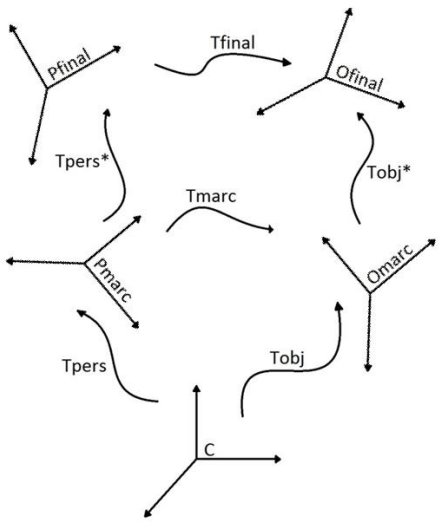
4 DESENVOLVIMENTO

Neste capítulo, descrevemos aspectos importantes no desenvolvimento dos protótipos. Nas Seções 4.1 e 4.2, discutimos aspectos relacionados com a colisão entre o objeto virtual e o personagem. Nas Seções 4.3, 4.4 e 4.5, discutimos detalhes relacionados mais especificamente a cada um dos protótipos: OpenCV, ARToolKit miniatura e ARTollKit tamanho real, respectivamente. E na Seção 4.6, descrevemos aspectos relacionados à realização dos testes.

4.1 Conversão entre sistemas de coordenadas

Considerando que uma caixa com múltiplos marcadores é usada, e todos eles devem ser mapeados em um mesmo objeto virtual, a consistência entre todos esses marcadores e seus respectivos objetos virtuais desenhados deve ser mantida. Além disso, temos que adicionar adequadamente a estrutura física equivalente ao objeto virtual na simulação para que a colisão com o personagem ocorra de forma coerente. Para adicionar essa estrutura física corretamente, temos que tomar o cuidado de definir a orientação e a posição do objeto virtual, obtido a partir do ARToolKit, no mesmo sistema de coordenadas em que o personagem está definido na simulação.

Figura 11 – Conversão entre sistemas de coordenadas.



Fonte: Criada pelo autor, (2015)

Note que a *webcam* real define um sistema de coordenadas correspondente à câmera virtual, usado como referência pelo ARToolKit. Assim, todo marcador define um sistema de coordenadas relativo a esse sistema de referência. Na Figura 11, $Tpers$ e $Tobj$ correspondem às matrizes de transformação relacionando o sistema de coordenadas da câmera e os sistemas de coordenadas correspondentes aos marcadores do personagem e do objeto, respectivamente. Conhecidas essas matrizes, a matriz de transformação responsável pela conversão entre os sistemas do personagem e do objeto, $Pmarc$ e $Omarc$, é dada por $Tmarc = Tpers^{-1} * Tobj$.

Transformações adicionais podem ser necessárias para ajustar melhor tanto o personagem quanto o objeto, ambos relativos aos seus respectivos marcadores. $Tpers^*$ e $Tobj^*$ correspondem a essas possíveis transformações adicionais, levando aos novos sistemas de coordenadas $Pfinal$ e $Ofinal$, respectivamente. Considerando essas transformações adicionais, a matriz de transformação responsável pela conversão entre os sistemas do personagem e do objeto, $Pfinal$ e $Ofinal$, é dada por $Tfinal = (Tpers * Tpers^*)^{-1} * (Tobj * Tobj^*) = Tpers^{*-1} * Tpers^{-1} * Tobj * Tobj^*$.

Note que a posição e a orientação adequadas do objeto, relativo ao sistema de coordenadas do personagem, $Pfinal$, podem ser obtidas a partir dessa matriz resultante $Tfinal$ e atualizadas na simulação física, de maneira consistente com o que é desenhado.

Vale ressaltar que simplesmente atualizar diretamente a orientação e a posição do marcador do objeto virtual, de acordo com a matriz de transformação $Tobj$, não resultaria no efeito desejado, já que $Tobj$ relaciona o sistema de coordenadas definido pelo marcador do objeto, $Omarc$, e o sistema de coordenadas da câmera.

4.2 Controle da posição e orientação do objeto

Calculada a matriz $Tfinal$, contendo a orientação e a posição do objeto virtual, podemos então usá-la para atualizar a estrutura física do objeto virtual, fazendo com que a colisão ocorra corretamente. Contudo, as taxas de atualização da simulação física e do ARToolKit são diferentes, ocasionando descontinuidades na interação entre o objeto virtual e o personagem. Na Figura 12, podemos observar o objeto virtual atravessando o personagem devido a essa descontinuidade.

Figura 12 – Descontinuidade na interação.

Fonte: Criada pelo autor, (2015)

Como a simulação física perde informações de posição e de orientação devido às taxas de atualização serem diferentes, a solução é adicionar recursos dentro da simulação para a própria simulação interpolar essas descontinuidades.

A nossa proposta é aplicar uma força na estrutura física para impulsioná-la, simulando o efeito de uma mola amortecida. Assumindo que uma extremidade da mola está presa na estrutura física e a outra está presa em uma posição diferente, o efeito da mola consiste em levar a estrutura física da sua posição atual de volta para sua posição desejada definida. Note que essa mola se encontra em repouso quando as posições atual e desejada coincidem. Essa força é calculada a partir da fórmula $F = Ks * (Pd - Pa) - Kd * (Va)$, onde Pa e Pd são as posições 3D atual e desejada, respectivamente, Ks e Kd são as constantes da mola e do amortecedor, respectivamente, e Va é a velocidade linear 3D atual. Ks é responsável por controlar a rigidez da mola, enquanto Kd controla o amortecimento do movimento, responsável por parar a estrutura. Caso contrário, a estrutura física continuaria oscilando porque não haveria perda de energia. Essa força é responsável apenas pela interpolação da posição, sendo também necessário tratar a interpolação da orientação.

O controle da orientação é baseado em uma ideia semelhante. Além de uma força, um torque também é aplicado na estrutura física, simulando o efeito de uma mola amortecida angular para realizar o ajuste da orientação. Esse torque é dado por $T = Ks * (\theta d - \theta a) - Kd * (\omega a)$, onde θa e θd representam as orientações atual e desejada, respectivamente, Ks e Kd são as constantes da mola e do amortecedor angulares, respectivamente, e ωa é a velocidade angular 3D atual. Devemos ter em mente que uma orientação 3D pode ser

representada de diversas formas, dentre elas: Ângulos de Euler, Eixo e Ângulo, Matrizes de Rotação, e Quatérnios. Escolhemos representar as orientações através de quatérnios por ser uma representação mais concisa e porque suas operações são mais eficientes. Note que a subtração entre ângulos mostrada na equação do torque acima corresponde à seguinte operação de quatérnios $T = Ks * (Qd * Qa^{-1}) - Kd * (\omega a)$. Note que o quatérnio resultante ainda deve ser convertido para a representação Eixo e Ângulo, e o deslocamento angular é obtido pela multiplicação entre o eixo (vetor 3D) e o ângulo (escalar) correspondentes. Essa conversão é feita para poder gerar algo consistente com o torque 3D.

4.3 OpenCV

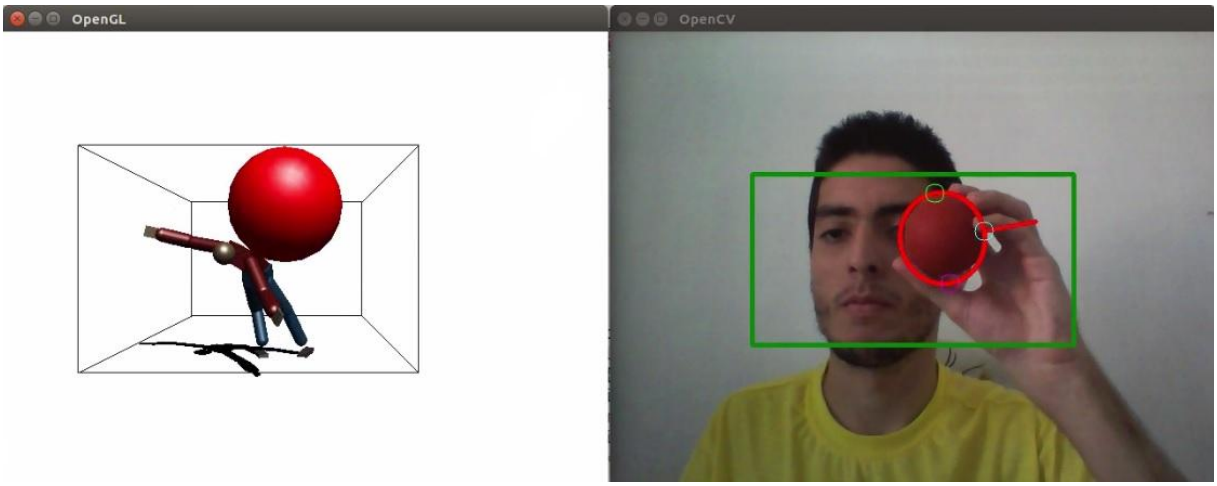
O OpenCV disponibiliza alguns códigos fontes com o uso de seus algoritmos como exemplo. Utilizamos o código fonte exemplo do CamShift como base, adaptando para o nosso uso e mesclando com o OpenGL.

Para o bom funcionamento do CamShift, é necessário a configuração manual de seus parâmetros matiz, saturação e brilho, antes de iniciar o rastreamento. Como o algoritmo faz o rastreamento com base na cor do objeto, preferimos não deixar esses parâmetros fixos para uma única cor. Em contra partida, sempre é necessário fazer a calibragem desses parâmetros. Para fazer esse calibragem, é necessário observar uma imagem *feedback* que é atualizada em tempo real com o objeto rastreado.

No OpenCV, a cada iteração, a imagem obtida da *webcam* é montada em uma matriz, matriz de imagem. Já no OpenGL, cada iteração é responsável por desenhar os objetos 3D. Como podemos unir o OpenCV com o OpenGL? A solução proposta neste trabalho é adicionar a função que monta a matriz de imagem do OpenCV dentro do *main loop* do OpenGL, dessa forma "sincronizando" as duas bibliotecas.

Com o OpenCV e o OpenGL integrados, criamos o ambiente virtual com o OpenGL, adicionamos o personagem fisicamente simulado e a estrutura física que interagem com ele no ambiente, e, a cada iteração, atualizamos a posição da estrutura física através dos dados fornecidos pelo OpenCV. O problema da descontinuidade citado acima também acontece nesse protótipo, contudo adicionamos somente o cálculo da força, pois nesse protótipo ignoramos a orientação do objeto 3D. A Figura 13 ilustra o funcionamento desse protótipo.

Figura 13 – Versão final do
protótipo OpenCV.



Fonte: Criada pelo autor, (2015)

4.4 ARToolKit Miniatura

A forma inicial de interação com o personagem era usando apenas um marcador, porém essa forma tornou-se inconveniente. Durante a realização de alguns testes específicos, posicionar o objeto para tocar no personagem tornou-se difícil, pois, ao realizar algumas rotações, o marcador deixava de aparecer na imagem da *webcam*. A solução encontrada foi a criação de uma caixa com cinco marcadores, cada um em uma face. Uma das faces foi retirada para que fosse possível colocar a mão e movimentá-la (Figura 14). A utilização dessa caixa tornou o manuseio do objeto virtual mais livre e com menos problemas de detecção, pois na maioria das vezes pelo menos um marcador se encontra bem posicionado em relação a *webcam*. Criamos cada marcador da caixa utilizando o método de criação explicado na Seção 2.2.4. A desvantagem do uso da caixa é o cuidado necessário para ajustar as transformações $Tobj^*$ (Seção 4.1) para cada face, para que exista coerência na troca de um marcador para outro.

Figura 14 – Caixa marcador.

Fonte: Criada pelo autor, (2015)

Na versão final dessa interação miniatura, substituímos o marcador que representava o personagem da versão inicial por um marcador específico da caixa. Além da liberdade de movimentar o sistema de coordenadas (base) do personagem, permitindo posicioná-lo em qualquer superfície do mundo real, também permitimos realizar rotações adicionais no eixo Y local dessa base via teclado. Ajustar as transformações $Tpers^*$ (Seção 4.1), para que o personagem ficasse coerente com a superfície, também foi trabalhoso. Mostrar a sombra do personagem e o chão virtual, parcialmente transparente, ajudou nessa tarefa. Nessa versão em miniatura, representamos o objeto virtual tanto através de uma cápsula quanto através de uma esfera. A mudança de um objeto para outro é feita apertando um botão. A Figura 15 ilustra o funcionamento desse protótipo.

Figura 15 – Versão final do protótipo ARToolKit miniatura.

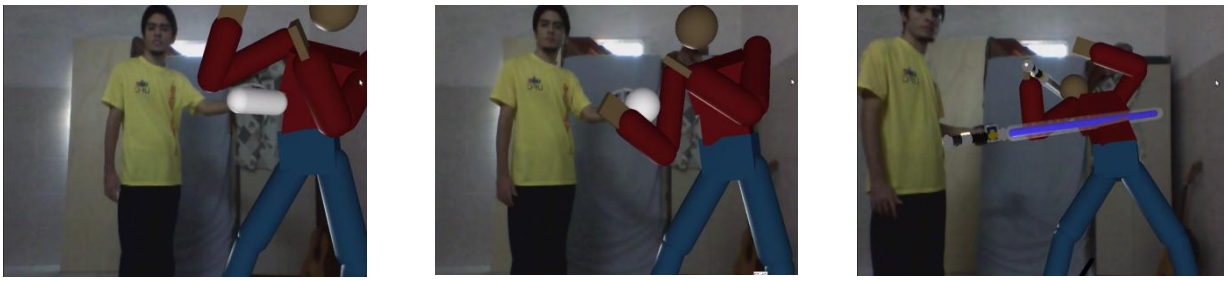
Fonte: Criada pelo autor, (2015)

4.5 ARToolKit Tamanho Real

Usando a mesma caixa citada na versão miniatura, partimos do mesmo princípio de usar um marcador para representar a base do personagem, porém, como temos que posicionar o personagem longe da *webcam*, não podemos escolher um marcador específico

para representar a base, pois a distância dificulta a detecção do marcador, e como foi dito anteriormente, na interação em miniatura o uso da caixa faz com que tenha sempre um marcador bem posicionado em relação a *webcam*. Dessa forma, deixamos todos os marcadores da caixa para representar o personagem. O problema de usar todos os marcadores para representá-lo é o fato de ser necessário ajustar as transformações $Tpers^*$ (Seção 4.1) para cada marcador. Como nesse protótipo permitimos alterar o tamanho do personagem em tempo de execução, em vez de ajustarmos todas essas transformações $Tpers^*$ antecipadamente, optamos por ajustá-las também em tempo de execução, pois elas são afetadas pelo fator de escala aplicado ao personagem. Assim como na versão em miniatura, após realizar adequadamente esses ajustes e pressionar um botão, a caixa passa a mapear um objeto virtual, que pode ser representado por uma cápsula, por uma esfera ou por um sabre de luz. A mudança de um objeto para outro também é feita apertando um botão. A Figura 16 ilustra o funcionamento desse protótipo em tamanho real, com as possíveis representações usadas para o objeto virtual.

Figura 16 – Versão final do protótipo ARToolKit tamanho real.



Fonte: Criada pelo autor, (2015)

4.6 Realização dos testes

Todos os protótipos, OpenCV, ARToolKit miniatura e tamanho real, usam visão computacional, seja para detectar os marcadores ou para fazer o rastreamento de um objeto. Para isso, é necessário usar algum dispositivo de captura de imagem. Em nosso caso, utilizamos uma *webcam* de 2 mega pixel (MP) como dispositivo de captura de imagem. Como

usamos um dispositivo simples de poucos MP, é muito importante o ambiente estar apropriado para a realização dos testes.

Nossos testes foram realizados em um ambiente que possuem as seguintes características:

- Ambiente fechado.
- Paredes brancas.
- 1 luz branca de 15W no teto a 3m de distancia do chão.

O computador usado nos testes e durante todo o desenvolvimento possui a seguinte configuração:

- Sistema operacional Ubuntu 14.04 LTS.
- Processador Intel Core i5 modelo 2450M com 2.5GHz.
- 6GB de memória RAM.
- Placa de video Intel HD Graphics 3000.

Antes do usuário testar cada protótipo, era dado uma explicação de como usar o sistema. Cada usuário ficou livre para interagir com o personagem fisicamente simulado à sua maneira, tendo apenas que seguir um roteiro inicial para se familiarizar com o sistema. Após a realização dos testes, os usuários fizeram um relatório descrevendo como foi usar cada protótipo.

5 DISCUSSÃO

Com base nos relatórios não podemos fazer afirmações gerais para todos os casos, porém podemos afirmar que todos os usuários que testaram o sistema sentiram-se imersos nos três protótipos. Os usuários relataram que os movimentos de reação do personagem eram previsíveis, assumindo o seguinte significado para "previsível": "Se fosse uma pessoa real sendo tocada pelos objetos usados, esse seria o resultado esperado". Podemos perceber que o usuário não estava esperando um conjunto fixo de animações de reação para o personagem. As expectativas eram observar reações mais contínuas como as de uma pessoa real. É importante também relatar que não teve um protótipo que se sobressaiu ao outro, pois todos possuíam seus pontos fortes e fracos. Todos os usuários gostaram dos três protótipos e, com base em seus relatórios, podemos destacar as seguintes características de cada protótipo.

- Protótipo OpenCV:
    - Permite realizar ações com maior velocidade sem perder a referência.
    - Não sentiu-se a falta de controlar a orientação do objeto virtual, pois era uma bola.
    - Não sentiu-se a necessidade de ficar olhando a imagem captada pela *webcam* para se orientar no ambiente virtual.
- Protótipo ARToolKit versão miniatura:
    - O personagem inserido em um ambiente real aparenta ser mais realista.
    - Aumenta a sensação de que realmente pode-se tocar no personagem.
    - A movimentação do objeto virtual no ambiente é maior em relação ao personagem. Com pequenos deslocamentos no marcador, é possível percorrer toda a extensão útil do ambiente real.
- Protótipo ARToolKit versão tamanho real:
    - O personagem inserido em um ambiente real aparenta ser mais realista.
    - É mais fácil tocar em um canto especifico do personagem.
    - Possui uma melhor visualização da reação do personagem.

Em contra partida devemos destacar que os usuários utilizavam a caixa com os marcadores sem se preocupar com a velocidade de seus movimentos. Isso fazia com que os marcadores ficassem fora de foco impedido a atualização adequada dos dados do objeto virtual. Tanto no protótipo miniatura quanto no protótipo tamanho real, esse problema acontecia. Para evitá-lo, era necessário realizar movimentos mais lentos. Podemos dizer que os protótipos usando ARToolKit possuem uma limitação na velocidade de movimentação dos marcadores, e não foi possível identificar o quanto essa limitação influenciou negativamente na imersão.

No OpenCV, ignoramos o controle da orientação do objeto virtual. Esse controle passou desapercebido pelos usuários pelo fato do objeto virtual ser unicamente uma esfera, surgindo a seguinte dúvida: “Se fosse um outro objeto, os usuários sentiriam a necessidade de controlar sua orientação? Se fosse uma cápsula igual aos protótipos usando ARToolKit, por exemplo, o que os usuários relatariam?”. Embora o OpenCV tenha apresentado uma precisão melhor no rastreamento, é necessário destacar como ponto fraco do protótipo do OpenCV os muitos ajustes iniciais que são necessários fazer para conseguir um bom rastreamento e uma boa movimentação nos eixos X, Y e Z.

6 CONSIDERAÇÕES FINAIS

Este trabalho partiu da possibilidade de interagir com um personagem fisicamente simulado, tendo como base um protótipo previamente implementado. Percebeu-se que essa implementação prévia possuía algumas limitações que não permitiam uma interação de maneira fácil. Uma pessoa leiga enfrentaria muita dificuldade para se adaptar e realizar as interações.

Trabalhamos em cima dessa implementação buscando melhorias, tendo sempre em mente a pergunta: “O que um usuário acharia desse protótipo?”. Como sempre tivemos a carência de saber qual a opinião de uma pessoa que não possuía ligação com esse trabalho, a realização dos testes era algo imprescindível para termos um *feedback* para futuras melhorias e possíveis correções de *bugs*.

Com a criação de um protótipo de realidade aumentada com tamanho miniatura, surgiu a ideia de aumentar o tamanho do personagem para deixá-lo com o tamanho de uma pessoa real. Seria também interessante desenvolver todo o cenário de um jogo. Por exemplo, um jogo de luta em que o usuário interage com um personagem em tamanho real seria bastante atrativo. Contudo, só tratamos a colisão do usuário com o personagem, e o recurso para o usuário interagir com o personagem é uma simples caixa. Se tentarmos comparar o uso dessa caixa com os recursos atuais para jogos, seria parecido com o *joystick* do *Nintendo Wii*. Note que apesar de todos os custos envolvidos na tecnologia usada nesse joystick, conseguimos desenvolver uma aplicação correspondente usando uma simples *webcam* e uma caixa feita com simples marcadores de papel, obtendo resultados que empolgaram os usuários.

Continuando as comparações com recursos de jogos existentes, o protótipo do OpenCV se encaixaria em uma comparação com o *Microsoft Kinect*. Enquanto nossa aplicação usando OpenCV ainda exige ajustes iniciais manuais para detectar um objeto real pré-determinado, o *Kinect* detecta todo o corpo do usuário de forma automática. Entretanto, apesar dos ajustes manuais, conseguimos um rastreamento satisfatoriamente preciso, inclusive da informação de profundidade, usando um algoritmo que não prevê diretamente essa informação.

Conseguimos realizar os nosso objetivos de criar os protótipos e avaliar a interação em todos casos previstos. Enfrentamos problemas interessantes durante o desenvolvimento e propomos soluções fáceis para cada problema. Entramos um pouco na simulação física mesmo não sendo a parte de interesse deste trabalho. Fugimos um pouco do

cronograma inicial determinado, mas isso foi um ponto necessário para conseguir todos os objetivos propostos para este trabalho. Trabalhar com computação gráfica é algo divertido, pois o desenvolvedor pode visualizar o que está fazendo.

Como trabalhos futuros, temos a ideia de posicionar o objeto virtual em relação à caixa com os marcadores de forma automática, buscar formas de diminuir as limitações encontradas nos protótipos e, com os *feedbacks* obtidos dos usuários, realizar os ajustes necessários nos protótipos. Também seria muito interessante migrar o código dos protótipos para alguma plataforma móvel.

REFERÊNCIAS

ARis. Disponivel em: <https://www.youtube.com/watch?v=yCCx7zANsGE/>. Acesso em 29 out 2014.

ARToolKit Marker Generator. Disponível em: <http://flash.tarotaro.org/blog/2009/07/12/mgo2/>. Acesso em 04 maio 2015.

ARToolWorks. Creating and training new ARToolKits markes. Disponível em: <https://www.artoolworks.com/support/library/Creating_and_training_new_ARToolKit_markers>. Accesso em 09 abr. 2015.

AZEVEDO, Eduardo; CONCI, Aura; LETA Fabiana R. **Computação gráfica: teoria e prática**. 2.ed. Rio de Janerio: CAMPUS, 2008. 432 p.

BIANCHI, Reinaldo AC; REALI-COSTA, Anna H. O Sistema de Visão Computacional do time FUTEPOLI de Futebol de Robôs. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 13., 2000, Florianópolis. **Anais...** Florianópolis: UFSC/Sociedade Brasileira de Automática, 2000, p. 2156-2162.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV: Computer vision with the OpenCV library**. New York: O'Reilly Media, 2008. 555 p.

ETIENNE, Jerome; Punch a Doom Character in Augmented Reality. Disponível em: <http://learningthreejs.com/blog/2012/05/15/punch-a-doom-character-in-augmented-reality/>. Acesso em 03 dez. 2013.

GEIJTENBEEK, Thomas; PRONOST, Nicolas. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. **Computer Graphics Forum**, New York: USA, v.31, n.8, p.2492-2515, dez. 2012.

KIM, Sinyoung; KIM, Yeonjoon; LEE, Sung-Hee. On Visual Artifacts of Physics Simulation in Augmented Reality Environment. In: INTERNATIONAL SYMPOSIUM ON UBIQUITOUS VIRTUALREALITY, 6, 2011, Washington. **Proceedings...** Washington: IEEE Computer Society, 2011. 60 p. 25-28.

KIRNER, Claudio; SISCOUTTO, Robson. **Realidade virtual e aumentada: conceitos, projeto e aplicações.** Porto Alegre: Sociedade Brasileira de Computação, 2007.

LAGANIÈRE, Robert. **OpenCV 2 computer vision application programming cookbook**. Birmingham: Packt Publishing, 2011. 306 p.

LEFFA, Vilson J. Interação simulada: Um estudo da transposição da sala de aula para o ambiente virtual. In: LEFFA, Vilson J. **A interação na aprendizagem das línguas.** 2. ed. Pelotas: Educat, 2006. p. 181-218.

LIU, C.; ZORDAN, V. Natural user interface for physics-based character animation. In: ALLBECK, J.; FALOUTSOS, P. (Ed.). **Motion in Games**. Springer Berlin Heidelberg, 2011, (Lecture Notes in Computer Science, v. 7060). p. 1–14.

MONTEIRO, Rafael; PS Vita recebe jogo de realidade aumentada PulzAR. Disponível em: <http://www.techtudo.com.br/jogos/noticia/2012/06/ps-vita-recebe-jogo-de-realidade-aumentada-pulzar.html>. Acesso em 12 dez. 2013.

NGUYEN, Nam et al. Performance capture with physical interaction. In: EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 5, 2010, Madrid. **Proceedings...** Madrid: Eurographics Association, 2010, 256 p. 189-195.

NUNES, Rubens F. et al. Simulando Reações Flexíveis em Movimentos Capturados. In: SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY, 10, 2008, João Pessoa. **Proceedings...** 2008, João Pessoa: SBC, 2008. 389 p. 241-250.

ODE. Open Dynamics Engine. Disponível em: <http://www.ode.org/>. Acesso em 12 dez. 2013.

OpenCV. Open Source Computer Vision. Disponível em: <http://opencv.org/about.html>. Acesso em 28 out. 2013.

OpenGL. Open Graphics Library. Disponível em: <http://www.opengl.org/about>. Acesso em 12 dez. 2013.

PulzAR™. Disponível em: <http://pt.playstation.com/psn/games/detail/item500873/PulzAR%E2%84%A2/ >. Acesso em 29 dez 2013.

SANTIN, Rafael; KIRNER, Claudio.ARToolKit: Conceitos e ferramenta de autoria colaborativa. In: SISCOUTTO, Robson; COSTA, Rosa. **Realidade Virtual e Aumentada:** Uma Abordagem Tecnológica. Porto Alegre: Sociedade Brasileira de Computação, 2008. p. 178-276.

SONG, Peng; YU, Hang; WINKLER, Stefan. Vision-based 3D finger interactions for mixed reality games with physics simulation. In: INTERNATIONAL CONFERENCE ON VIRTUAL-REALITY CONTINUUM AND ITS APPLICATIONS IN INDUSTRY, 7, 2008, Singapore. **Proceedings...** New York: ACM, 2008. 223 p. 1-6.

TORI, Romero; KIRNER, Claudio; SISCOUTTO, **Robson Augusto. Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Sociedade Brasileira de Computação, 2006, p. 412.

APÊNDICES

APÊNDICE A – Instalação ARToolKit no Ubuntu 14.04LTS

1 - Instalação das libs abaixo:
sudo apt-get install freeglut3-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libxi-dev libxmu-headers libxmu-dev libjpeg62-dev libglib2.0-dev libgtk2.0-dev

2 - Extraia ARToolKit-2.72.1.tgz

3 - Abra o arquivo Configure que estar dentro da pasta ARToolKit, e substitua o conteúdo da linha 115 pelo contudo abaixo:
LIBS="-lpthread -lglut -lGLU -lGL -lXi -lX11 -lm $GST_LIBS"

4 - Usando o terminal vá para a pasta do ARToolKit e digite os seguintes comandos:
./Configure

Vai aparecer

"1: Video4Linux" para capturadoras de video (mediante V4L versión 1).
"2: Video4Linux+JPEG Decompression (EyeToy)" para cámaras Play Station EyeToy (mediante V4L versión 1).
"3: Digital Video Camcoder throught IEEE 1394 (DV Format)" para cámaras firewire.
"4: Digital Video Camera throught IEEE 1394 (VGA NONCOMPRESSED Image Format)" para cámaras firewire.
"5: GStreamer Media Framework" para webcams USB.

escolha a opção 5

Vai aparecer

Do you want to create debug symbols? (y or n)

escolha y

Vai aparecer

Build gsub libraries with texture rectangle support? (y or n)
GL_NV_texture_rectangle is supported on most NVidia graphics cards and on ATi Radeon and better graphics cards

escolha y

depois digite make

espere o processo acabar demora alguns minutos... se tudo ocorrer bem execute os seguintes comandos:

```
sudo cp -R ./include/AR /usr/local/include/

sudo cp ./lib/*.a /usr/local/lib/
```

agora crie um arquivo com o nome "AR.pc" (sem aspas) e cole o conteudo abaixo nele:

```
prefix=/usr/local
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${exec_prefix}/include


Name: AR
Description: ARToolKit libs and includes
Version: 2.72.1
Libs: -L${libdir} -lARgsub -lARgsub_lite -lARgsubUtil -lARMulti -
lARvideo -lAR
Cflags: -I${includedir}/AR
```

salve o arquivo e digite o comando no terminal:

```
sudo cp AR.pc /usr/lib/pkgconfig/
```

execute o comando:

```
pkg-config --cflags --libs AR
```

se aparecer:

```
-I/usr/local/include/AR  -L/usr/local/lib -lARgsub -lARgsub_lite -
lARgsubUtil -lARMulti -lARvideo -lAR
```

significa que deu tudo certo.

lembre-se de quando for fazer um projeto usado a biblioteca ARToolKit, configurar a variavel *vconf com o conteudo abaixo:

```
char *vconf = "v4l2src device=/dev/video0 use-fixed-fps=false !
videoflip method=horizontal-flip ! ffmpegcolorspace ! capsfilter
caps=video/x-raw-rgb,bpp=24,width=640,height=480 ! identity
name=artoolkit ! fakesink";
```

Para quem usa o Qt Creator é necessario configurar o arquivo .pro adicionando:

```
LIBS += `pkg-config AR --cflags --libs`
```

APÊNDICE B – Roteiro inicial

Atingir com a extremidade do bastão/cilindro o braço esquerdo do personagem.

Bater o bastão de lado na cabeça do personagem, com uma certa velocidade.

Dar um soco (empurrar/deslocar) no ombro direito do personagem

Levantar um pé do personagem

Curvar a coluna do personagem para baixo

Atirar exatamente na cabeça do personagem

Empurrar o abdômen do personagem

APÊNDICE C – Questionário inicial

Você se sentiu imerso no ambiente?

Quais aspectos do sistema ajudam na sensação de imersão no ambiente?

Quão previsíveis são as reações do personagem? Você percebeu que ele é fisicamente simulado (sua animação envolve física)?

Como isso ajuda na qualidade de interação (na sensação de imersão)?

Qual forma de interação é melhor (mais natural?) Posicionar diretamente o objeto/"marcador" ou usar molas(colisão mais precisa)?

Que benefícios o controle da orientação do objeto/"marcador" proporciona à interação (sensação de imersão)?

Qual o melhor objeto/"marcador" para interagir? Em que situação cada um se comporta melhor? Ou um deles é melhor em todos os casos? (esfera vs bastão/capsula vs lightsaber)

Qual biblioteca permitiu uma melhor interação? Bolinha vermelha ou o personagem no ambiente (miniatura e tamanho real)

Como a realidade aumentada influenciou na sensação de imersão?

Tente descrever um pouco como foi a interação com cada um dos protótipos.